

GUIDE DE CONCEPTION
DU DOSSIER PROJET

TITRE PROFESSIONNEL
DE CONCEPTEUR
DEVELOPPEUR
D'APPLICATION

WASSIM BACHA

2024



Uber Cut

Cahier des charges & Expression des besoins

Table des matières

Personal presentation	4
About me.....	4
About my job	4
ADventori.....	4
DCO	4
Mon rôle	5
Une de mes missions	5
Conclusion	5
Présentation du projet	6
Cas d'un utilisateur.....	6
Cas d'un coiffeur	6
Cahier des charges.....	6
Objectifs	6
Cibles.....	7
Exigences fonctionnelles	7
Exigences 'Front-end'	7
Exigences 'Back-end'	8
L'utilisateur.....	9
Confidentialité	9
Droits d'accès.....	10
Exigences & choix techniques	10
Exigences	10
Choix	10
Définition du MVP (Minimum Viable Product)	11
Contrainte de temps	12
Méthodologie & Organisation.....	12
Conception de l'interface graphique	16
Charte graphique.....	16
Les couleurs	17
Le logo	17
Le Wireframe	17
Maquettage	18
Conception de la base de données	19
Modèle Conceptuel de Données (MCD)	19
Modèle Logique de Données (MLD)	21

Conception de l'application.....	23
Diagramme de Cas d'Utilisation (Use Case).....	23
Conception multicouche MVC.....	25
Le Modèle.....	25
La Vue	28
Le Contrôleur	29
Communication entre les 3 composants	31
L'architecture 3 tiers.....	33
Sécurité	35
Les attaques XSS (Cross-Script Scripting).....	35
Les injections SQL	36
Les attaques CSRF (Cross-Site Request Forgery)	37
Les tests unitaires	38
Veille Technologique	40
Veille globale	40
Veille sécurité.....	42
Difficultés rencontrées	44
Remerciements.....	45
Conclusion	46

Personal presentation

About me

Hello, I'm Wassim BACHA, currently 22 years old. I'm in my third year of full-stack development studies at EFREI while working at ADventori.

Although I currently reside in Villejuif, I've had the opportunity to move quite frequently, exploring different cities and opportunities. Last year, I worked as a Computer Engineer at Hanover Displays in Nice.

I'm deeply passionate about software development and have a keen interest in video games. After completing my baccalauréat in Sciences, I pursued a multidisciplinary academic path in mathematics, computer science, and physics, followed by two years in a BTS SIO SLAM program in alternating work-study format. Today, I'm nearing the completion of my third year in a bachelor's program specializing in web and application development, still in alternating work-study format.

The project I'm presenting today is called Uber Cut. It's an innovative platform for scheduling appointments with local hairdressers.

About my job

ADventori

ADventori est une start-up créée en 2009 par Pierre-Antoine Durgeat, anciennement créateur de l'application Mappy.

Nous travaillons dans le domaine du marketing digital où nous sommes leader en France des bannières de publicité dynamiques (DCO).

DCO

La DCO (Dynamic Creative Optimisation) est une technologie publicitaire qui permet de personnaliser en temps réel les informations présentes sur les bannières publicitaires.

Le contenu d'une bannière publicitaire s'adapte à l'utilisateur et à son contexte de navigation (profil, géolocalisation, flux de produits, météo, etc...).

Les clients peuvent diffuser une infinité de publicités à partir d'une seule bannière publicitaire.

Tout cela est possible grâce à du code, principalement en Groovy (Java) & JavaScript, sur lesquels j'ai été formé cette année.

Mon rôle

Au sein d'ADventori, je suis Développeur Fullstack Junior et mon but est de m'occuper des tickets sur lesquels je suis attribué.

Cela me permet de pouvoir travailler sur une large sélection de sujets avec des stack et des technologies différentes.

Mes missions varient et consistent principalement à réparer du code, ajouter des fonctionnalités ou encore optimiser du code.

Une de mes missions

Une de mes missions à été de développer avec mon tuteur une Creative Management Platform (CMP) pour le client Renault.

Il s'agit d'un dashboard permettant au client de monitorer, déployer et modifier ses bannières publicitaires.

Le langage de développement était Groovy (Java) et la mission nous à pris entre 5 et 10 jours à compléter.

Conclusion

ADventori est un très bon tremplin pour mon apprentissage dans le développement fullstack, j'ai appris énormément de méthodologies de travail et forgé de solides bases en développement.

Mon but pour la suite serait donc de monter en compétences et de pouvoir exercer plus de responsabilités au sein de l'entreprise.

Par la suite, je pourrais donc étudier en mastère de développement Fullstack & Manager tout en passant à l'étape supérieure et pouvoir approfondir mes connaissances en développement, car notre solution est nourrie de plus de 10 ans de développement dans tous types de langages et toutes sortes de technologies.

Présentation du projet

A l'origine, Uber Cut était une idée que j'ai eu en 2020 lors de la crise sanitaire du Covid-19, date à laquelle il était très compliqué de trouver un rendez-vous chez le coiffeur.

Uber Cut est donc une application web permettant à des clients de réserver des prestations à domicile.

Il existe 2 cas principaux d'utilisation.

Cas d'un utilisateur

Dans le cas d'utilisation d'un client, celui-ci a la possibilité de se créer un compte et de réserver un coiffeur à domicile en remplissant un formulaire renseignant les diverses informations sur la coupe voulue.

À la suite du paiement, un rendez-vous est créé.

Le coiffeur effectue son travail et le client a ensuite la possibilité d'envoyer une note sur 5 étoiles accompagnée d'un commentaire.

Cas d'un coiffeur

Dans le cas d'utilisation d'un coiffeur, celui-ci a la possibilité de se créer un compte, de renseigner ses diplômes, ajouter une photo de profil, une description et de se marquer comme 'prêt à coiffer' sur la plateforme dédiée sur l'application web.

Lorsqu'un client réserve un rendez-vous avec lui, le coiffeur a le choix de refuser ou d'accepter et d'aller à l'adresse du client pour lui faire sa prestation.

Ensuite, le coiffeur recevra son paiement.

Cahier des charges

Voici quelques points importants de la partie fonctionnelle de l'application :

Objectifs

Uber Cut est une plateforme web gratuite permettant de mettre en relation les clients et les coiffeurs. Le but est de permettre à n'importe qui de profiter du confort de se faire coiffer à domicile.

Uber Cut est aussi conçue afin d'être une opportunité de travail pour les coiffeurs amateurs et professionnels.

Les clients pourront réserver des coiffures à domicile et paieront plus ou moins cher dépendant de la coupe demandée et des compétences du coiffeur choisi.

Un système de notation permettra aux coiffeurs talentueux d'évoluer en étant mis en avant et en profitant de taxes réduites.

Cibles

Les cibles de l'application Uber Cut sont répartis en 2 groupes :

Les clients :

- Les particuliers souhaitant bénéficier de services de coiffure à domicile.
- Les personnes à mobilité réduite ne pouvant pas se déplacer facilement jusqu'à un salon de coiffure.

Les Coiffeurs :

- Les professionnels de la coiffure afin d'arrondir leur fin de mois en offrant des services à domicile.
- Les débutants de la coiffure souhaitant apprendre tout en se faisant de l'argent de poche.

Exigences fonctionnelles

Ici seront présentées les exigences fonctionnelles de l'application Uber Cut.

Exigences 'Front-end'

- **Page d'accueil**

La page d'accueil doit être présente pour fournir un aperçu des services de l'application et permettre une navigation facile vers les autres sections.

- **Page Profil**

La page profil doit permettre aux utilisateurs et aux coiffeurs de visualiser et de modifier leurs informations personnelles, telles que leurs coordonnées ainsi que leur historique de commandes et leur facture.

- **Page Administration**

La page d'administration doit être disponible pour que les administrateurs puissent gérer les utilisateurs, les coiffeurs, les réservations et autres aspects de la plateforme.

- **Page Dashboard Coiffeur**

La page dashboard dédiée aux coiffeurs doit leur permettre de gérer leur disponibilité, d'accepter ou de refuser des rendez-vous et de modifier leur vitrine de coiffeur (description & coupes disponibles).

- **Page Dashboard Client**

La page dashboard dédiée aux clients doit leur permettre d'effectuer une recherche de coiffeurs dans les environs à partir d'un formulaire.

Ils peuvent ainsi sélectionner un coiffeur et créer un rendez-vous.

- **Page Historique des commandes**

La page historique des commandes doit lister toutes les réservations passées des utilisateurs, avec les détails des prestations et la facture correspondante.

- **Page Mentions légales**

La page "Mentions légales" doit être présente pour fournir les informations juridiques et réglementaires relatives au site.

Exigences 'Back-end'

- **Gestion des utilisateurs**

Implémenter un système sécurisé d'inscription et d'authentification pour les utilisateurs et les coiffeurs tout en gardant la possibilité de modifier ou supprimer son profil.

- **Gestion des réservations**

Permettre aux utilisateurs de créer des rendez-vous et aux coiffeurs de mettre à jour leur disponibilité en temps réel.

- **Evaluation & Commentaires**

Permettre aux utilisateurs de laisser des notes et des commentaires sur les prestations des coiffeurs.

- **Dashboard Administrateur**

Gérer les droits d'accès et les rôles des différents utilisateurs (administrateurs, modérateurs, coiffeurs, clients).

- **Sécurité**

Assurer la protection des données personnelles des utilisateurs et des coiffeurs.

L'utilisateur

Une fois que l'utilisateur a créé son compte, il doit avoir accès au contenu du site et à la page de modification du profil.

Les fonctionnalités disponibles aux utilisateurs sont :

- Consultation/Modification/Suppression de son compte
- Recherche de coiffeurs dans les environs
- Réservation de Séances
- Consultation de ses Réservations
- Évaluation des Prestations
- Consultation des Factures

Si l'utilisateur est un coiffeur, à la création de son compte, il devrait avoir accès à la page de modification du profil ainsi qu'au dashboard de coiffeur.

Les fonctionnalités disponibles aux coiffeurs sont :

- Consultation/Modification/Suppression de son compte
- Gestion de la Disponibilité
- Gestion des Réservations
- Consultation de son Historique de Prestations
- Mise à Jour de la Vitrine

Confidentialité

Il est essentiel de protéger l'accès en lecture ou en écriture aux informations "privées" de chaque utilisateur. Les mesures suivantes doivent être mises en place :

- **Protection des Informations Privées** : Les informations privées des utilisateurs, telles que leurs réservations, leurs évaluations, et leurs données personnelles, doivent être strictement accessibles uniquement par eux-mêmes.
- **Suppression de Compte** : Les utilisateurs doivent pouvoir supprimer leur compte à tout moment, par exemple en cas de dysfonctionnement.
- **Visibilité et Modification des Données Privées** : Les réservations, commentaires, et évaluations créées par un utilisateur doivent être protégés de

la vue ou de la modification par les administrateurs du site ou d'autres utilisateurs qui ne sont pas directement concernés. Seuls les coiffeurs impliqués dans une réservation ou les utilisateurs spécifiques désignés par le créateur du contenu (par exemple, ajoutés à une liste de favoris ou d'amis) peuvent avoir accès à ces données privées.

Droits d'accès

L'accès aux différentes fonctionnalités et informations du site sera contrôlé en fonction des rôles du profil de l'utilisateur.

Les droits d'accès (lecture, écriture) doivent être déterminés en fonction de la catégorie d'informations et de l'identité de l'utilisateur.

Ces règles garantissent que les droits d'accès sont correctement gérés et que les informations sensibles des utilisateurs sont protégées tout en permettant une gestion efficace de la plateforme Uber Cut.

Exigences & choix techniques

Exigences

Afin de pouvoir atteindre le plus largement possible les cibles du projet, il conviendra que l'application soit fonctionnelle sur les principaux navigateurs internet, à savoir : Chrome, Firefox et Edge à minima.

En plus d'être adaptée aux écrans d'ordinateur, l'application devra être responsive, avec un accent particulier sur les appareils de type smartphone. Cette décision découle de la cible spécifique de l'application, qui vise principalement les utilisateurs mobiles.

Il est crucial de garantir une expérience utilisateur optimale sur les smartphones, principaux dispositifs utilisés pour accéder aux services de coiffure à domicile.

Le focus sera donc mis sur le développement d'une interface conviviale et adaptée aux smartphones, afin de permettre une utilisation aisée et intuitive pour les utilisateurs ciblés.

Choix

Afin de faciliter la mise en place du design responsive, on utilisera la librairie CSS Bootstrap permettant de créer des designs rapidement et efficacement.

Du point de vue des langages de programmation, on optera pour le framework Symfony PHP pour le Back-end.

Pour le Back-end (Symfony PHP) :

- La structure et l'organisation de Symfony à laquelle je suis déjà habitué, l'organisation y est claire et organisée facilitant le développement, la maintenance et la scalabilité du projet.
- Les fonctionnalités de sécurité intégrées telles que l'authentification, la gestion des droits d'accès et la protection contre les attaques courantes (CSRF, XSS).
- La large communauté de développeurs et la grande quantité de documentation, rendant plus facile la résolution des problèmes et l'accès à certaines informations.

Pour le Front-end (Twig & Bootstrap CSS) :

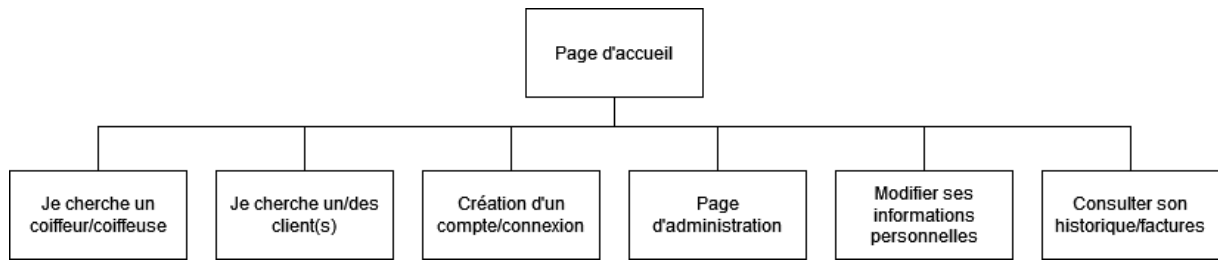
- Bootstrap permet de créer des designs responsifs rapidement et efficacement, assurant une interface utilisateur conviviale sur tous les types d'appareils.
- L'intégration de Twig avec Bootstrap permet de séparer clairement la logique du backend et la présentation, simplifiant ainsi le développement et la maintenance du front-end.

Pour mon IDE (Integrated Development Environment), j'ai fait le choix de Visual Studio Code pour sa simplicité d'utilisation, sa rapidité et son grand nombre d'addons de la communauté permettant de personnaliser son environnement de travail.

Définition du MVP (Minimum Viable Product)

Le MVP de l'application nous permettra :

- Inscription via un formulaire dans la page /register
- Connexion via un formulaire dans la page /login
- Effectuer une recherche de coiffeurs
- Effectuer une recherche de clients à coiffer
- Accès à la page d'administration pour les utilisateurs munis du rôle `USER_ADMIN`
- Modifier ses informations personnelles
- Consulter l'historique des commandes ainsi que leur facture.



Contrainte de temps

L'application est en cours de développement et devrait sortir publiquement d'ici janvier 2025.

Ce deadline me permettra ainsi de développer et peaufiner chaque élément présent sur le Diagramme de Use Case.

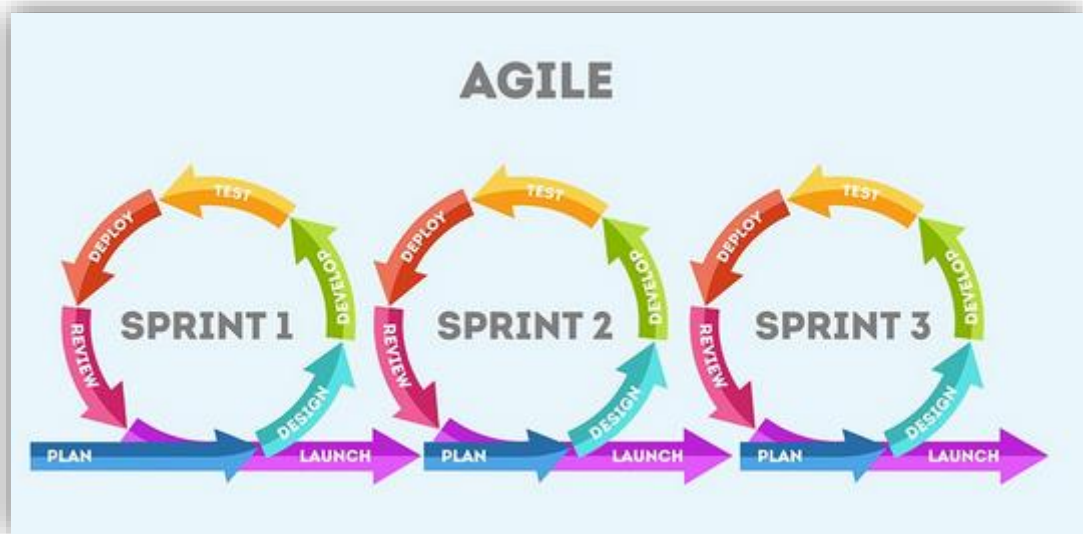
Cependant, étant donné le démarrage tardif du projet et les dates de rendu prévues du CDA, je me dois de proposer une application remplissant à minima les fonctionnalités du MVP.



Méthodologie & Organisation

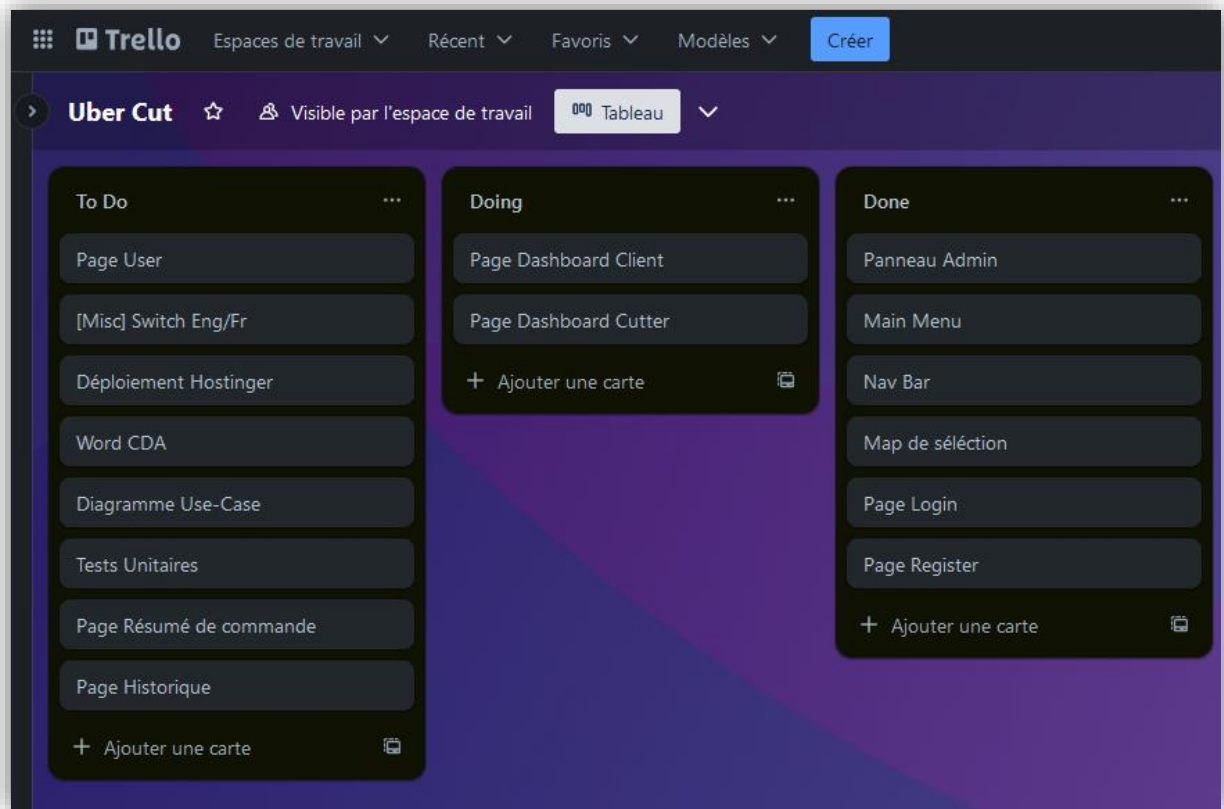
Dans le cadre de ce projet, j'ai choisi la méthode agile pour la simple raison qu'il s'agit d'un projet 100% autonome et que je me suis familiarisé avec cette méthode de travail au sein de mon alternance chez ADventori.

La méthode Agile est un ensemble de pratiques et de principes pour la gestion de projets et le développement de logiciels qui mettent l'accent sur la flexibilité et la satisfaction du client. Cette méthode se distingue par son approche itérative et incrémentale, permettant de livrer des parties fonctionnelles du produit à intervalles réguliers, d'incorporer les retours et de s'adapter rapidement aux changements.



Pour l'organisation de mes tâches dans le projet, je me suis servi de Trello où j'ai réparti les différents points d'amélioration de l'application sous forme de tickets rangés par ordre d'importance.

Cette méthode permet d'avoir une vue claire sur l'avancement du projet.



Pour le versionning du code j'ai utilisé un repository GitHub (Git) me permettant de garder à jour mon environnement de travail même lorsque je change de machine.

GitHub me permet aussi de garder un œil sur les versions précédentes du code afin de revenir en arrière si un problème survient.

Le lien du repository est disponible à l'adresse suivante :

<https://github.com/abwii/HubertCut>

The screenshot shows a GitHub repository named 'HubertCut' by user 'abwii'. The repository is public and has 32 commits. The commit history table lists the following changes:

Commit Message	Time
Pptx updated + added CutterCuts field	8 hours ago
working project, registration & login page added	last month
Added security on pages, password repetition & RGPD o...	last week
Pptx updated + added CutterCuts field	8 hours ago
fixed css on main page	last week
Pptx updated + added CutterCuts field	8 hours ago
added policy btn, removed "remember me" option	last week
[WIP] Added translation feature	last month
working project, registration & login page added	last month
working project, registration & login page added	last month
Update README.md	last month
working project, registration & login page added	last month
working project, registration & login page added	last month
Added admin page, CRUD & changes to register/login p...	last month
Added admin page, CRUD & changes to register/login p...	last month
Added admin page, CRUD & changes to register/login p...	last month

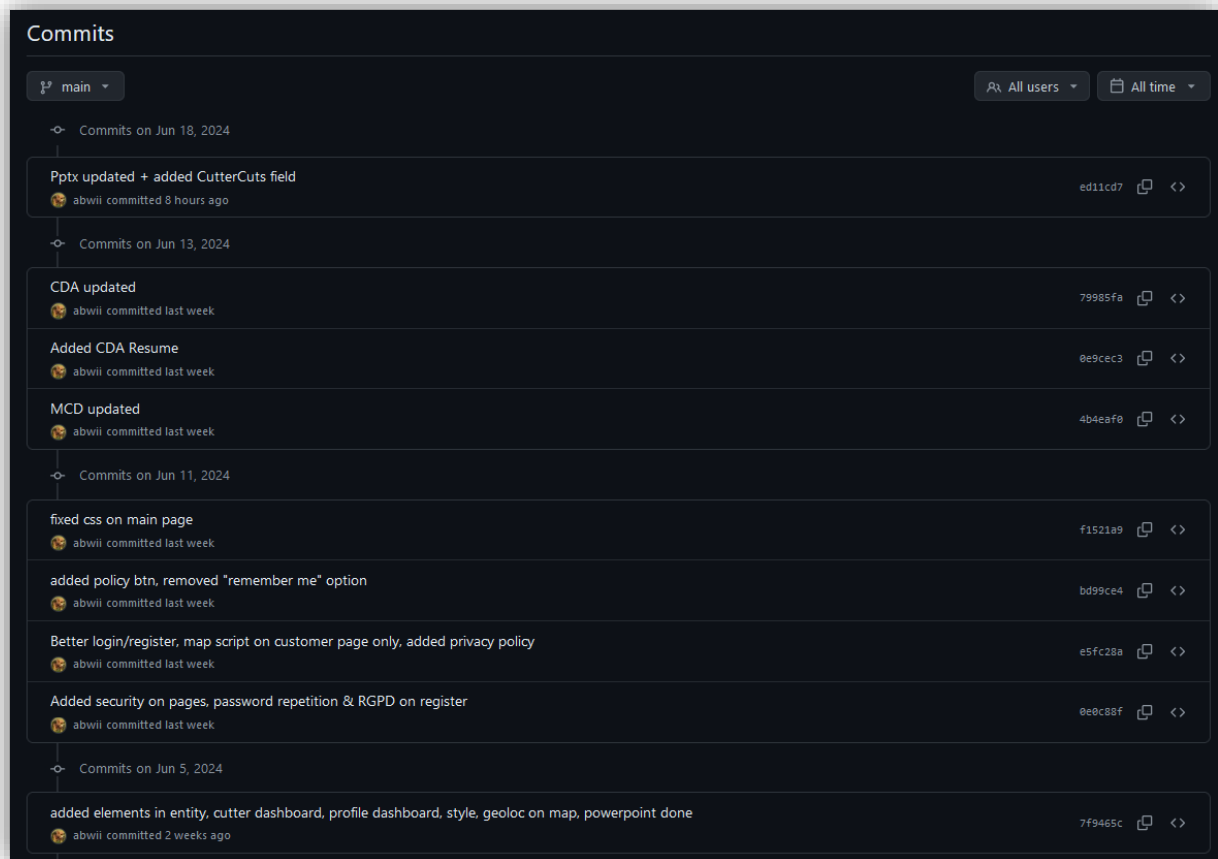
The file list on the right includes: _redac, bin, config, migrations, public, src, templates, translations, .env, .gitignore, README.md, compose.override.yaml, compose.yaml, composer.json, composer.lock, and symfony.lock.

On the right sidebar, the 'About' section states: 'No description, website, or topics provided.' The 'Releases' section says: 'No releases published. Create a new release.' The 'Packages' section says: 'No packages published. Publish your first package.' The 'Languages' section shows a bar chart with the following data:

Language	Percentage
PHP	59.3%
Twig	31.2%
CSS	5.0%
JavaScript	4.5%

The 'Suggested workflows' section is based on the tech stack and includes a 'Configure' button for PHP.

Voici un aperçu de mon historique de « Commits » sur GitHub, chaque ligne correspond à des changements dans le code.



Conception de l'interface graphique

La conception de l'interface utilisateur (UI) et de l'expérience utilisateur (UX) occupent un rôle primordial dans le développement d'une application ayant pour but d'être rapide, efficace et utilisable depuis des appareils à plusieurs formats d'écran.

Ce chapitre se concentre donc sur la phase de conception graphique.

L'aspect visuel, l'ergonomie et l'expérience utilisateur ont été les points les plus soigneusement conçus lors du développement de cette application.

Le public est habitué à des applications rapides, fluides et intuitives.

Dans l'objectif de démocratiser Uber Cut, il est donc essentiel de proposer une expérience satisfaisante aux utilisateurs.

Charte graphique

Afin de proposer une application visuellement agréable, je me suis grandement inspiré des applications Uber et Uber Eats dans les couleurs, la police d'écriture et l'agencement du site.

Les couleurs

La couleur principale du site est : #1f1f1f

La couleur secondaire du site est un blanc pur : #ffffff

Le logo

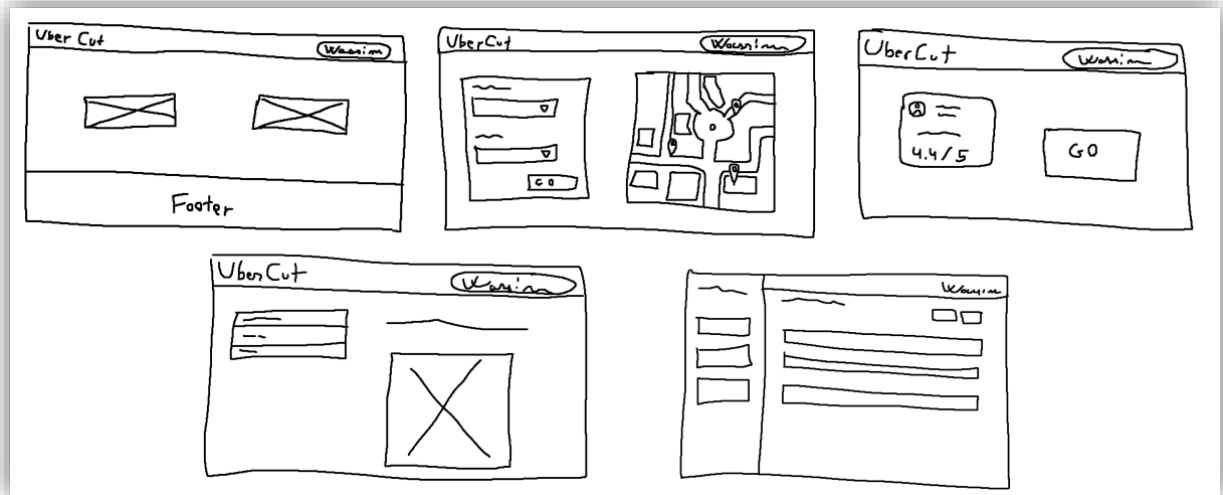
Pour le logo, il fallait quelque chose de simple et facilement reconnaissable, j'ai donc fait un cercle noir avec un contour blanc sur lequel j'ai placé une paire de ciseaux en blanc.



Le Wireframe

Un wireframe est une esquisse ou un schéma visuel qui représente la structure et la disposition d'une page web ou d'une application. Il s'agit d'un outil de conception essentiel utilisé au début du processus de développement pour définir la structure et le contenu des pages, sans se concentrer sur les détails visuels ou graphiques.

Voici les wireframes de l'application Uber Cut :

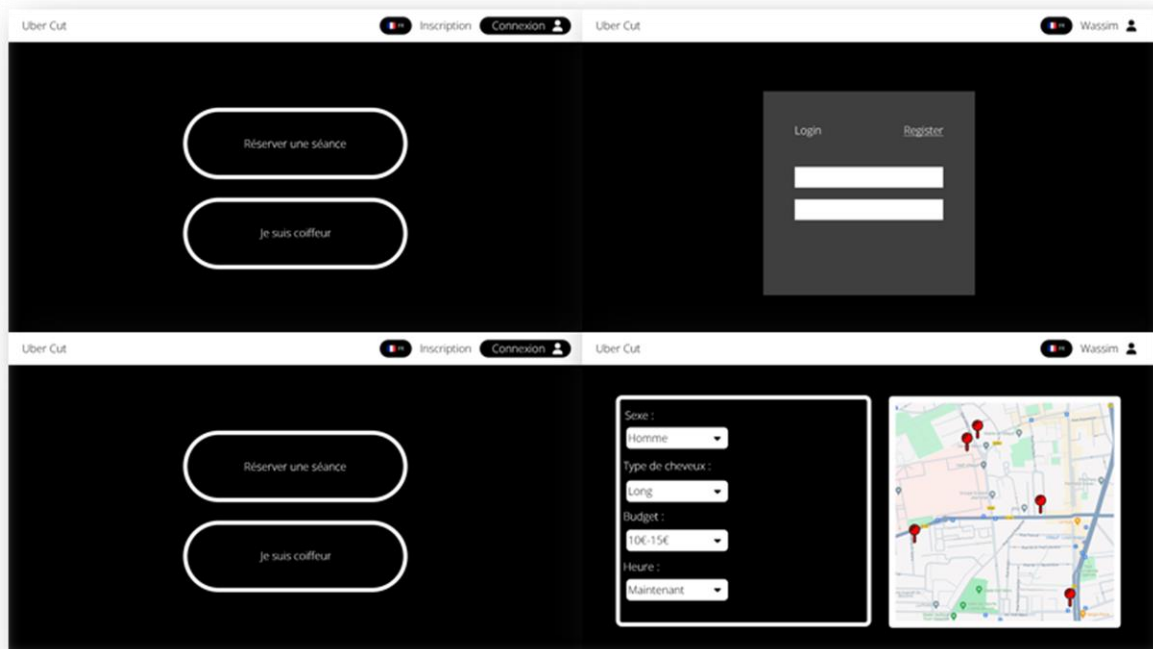


Maquettage

Le maquettage est une étape du processus de conception d'un site web et d'une application. Contrairement aux wireframes, qui se concentrent sur la structure et la disposition, les maquettes fournissent une représentation visuelle détaillée du produit final, incluant les couleurs, la typographie, les images et les éléments graphiques.

Pour Uber Cut, les maquettes garantiront une interface utilisateur intuitive et esthétiquement plaisante, offrant une expérience utilisateur optimale sur tous les appareils.

Voici la maquette de l'application Uber Cut :



Conception de la base de données

Structurer la base de données d'une application est un processus complexe mais essentiel pour éviter un développement à l'aveugle. En effet, cette structuration permet de comprendre les données que l'application utilisera. Elle aide à modéliser et représenter les entités, les relations et les attributs clés nécessaires au bon fonctionnement de l'application.

Pour cette étude, j'ai utilisé la méthode Merise, une approche de conception et de modélisation des systèmes d'information. La méthode Merise comprend plusieurs étapes clés : l'analyse des besoins, la modélisation conceptuelle des données (MCD), la modélisation logique des données (MLD) et la modélisation physique des données (MPD). Grâce à cette méthode, il est possible de concevoir des bases de données de manière rigoureuse et structurée.

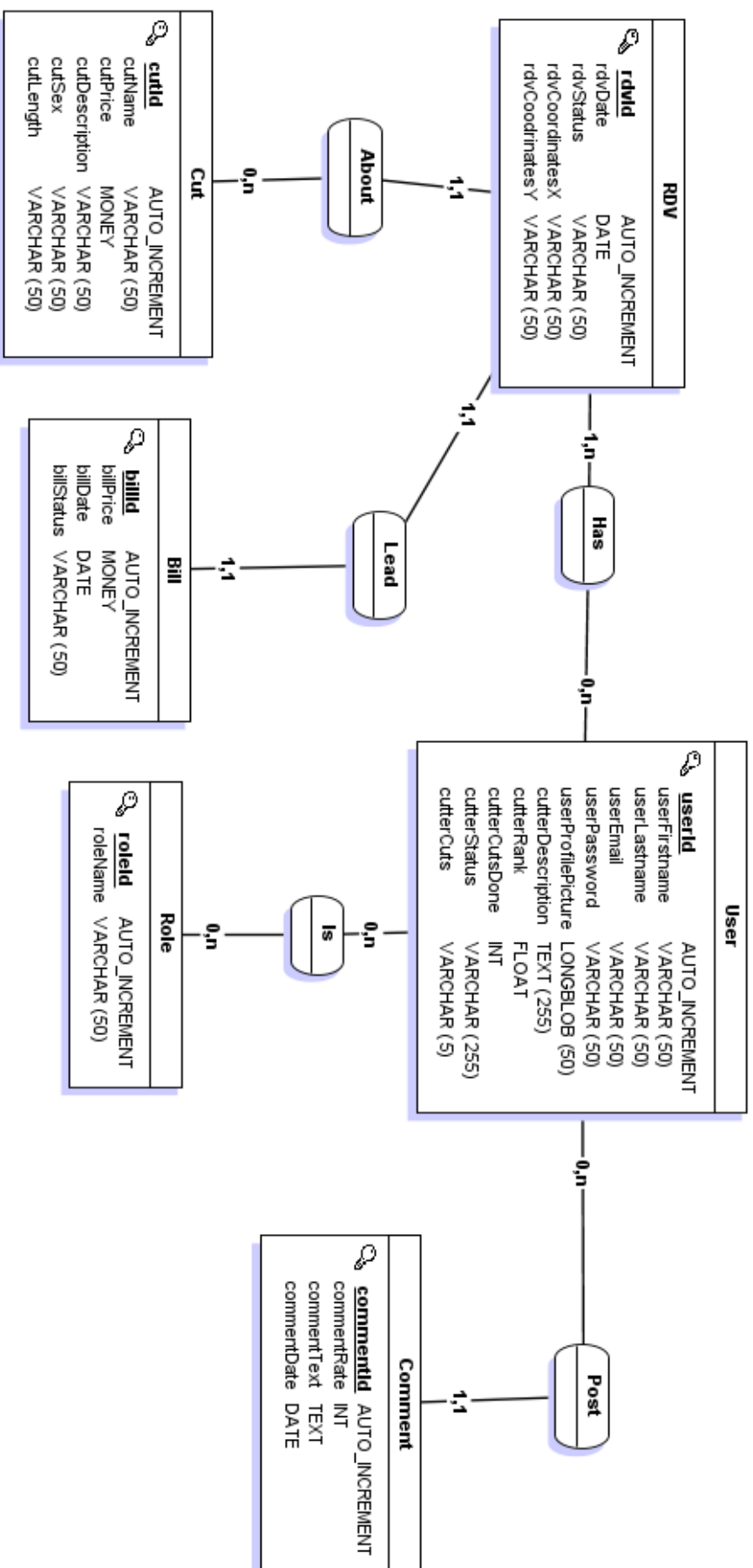
Modèle Conceptuel de Données (MCD)

Le MCD, ou Modèle Conceptuel de Données, est une représentation abstraite des concepts et des relations entre les entités d'un système d'information. Ce schéma conceptuel décrit les principales entités, leurs attributs et les associations entre elles, tout en étant indépendant de toute considération technique ou de mise en œuvre.

Le MCD utilise des symboles graphiques tels que des rectangles pour représenter les entités et des lignes pour représenter les relations entre ces entités. Les attributs des entités sont spécifiés à l'intérieur des rectangles, et les cardinalités des relations sont indiquées avec des notations appropriées.

L'objectif principal du MCD est de fournir une vue claire et abstraite des données du système, en mettant l'accent sur la structure logique et les relations entre les entités. Il permet de comprendre les besoins métier, d'identifier les entités clés, d'organiser les données de manière cohérente et de préparer la modélisation logique des données pour les étapes ultérieures de développement.

Voici le diagramme MCD de l'application Uber Cut :

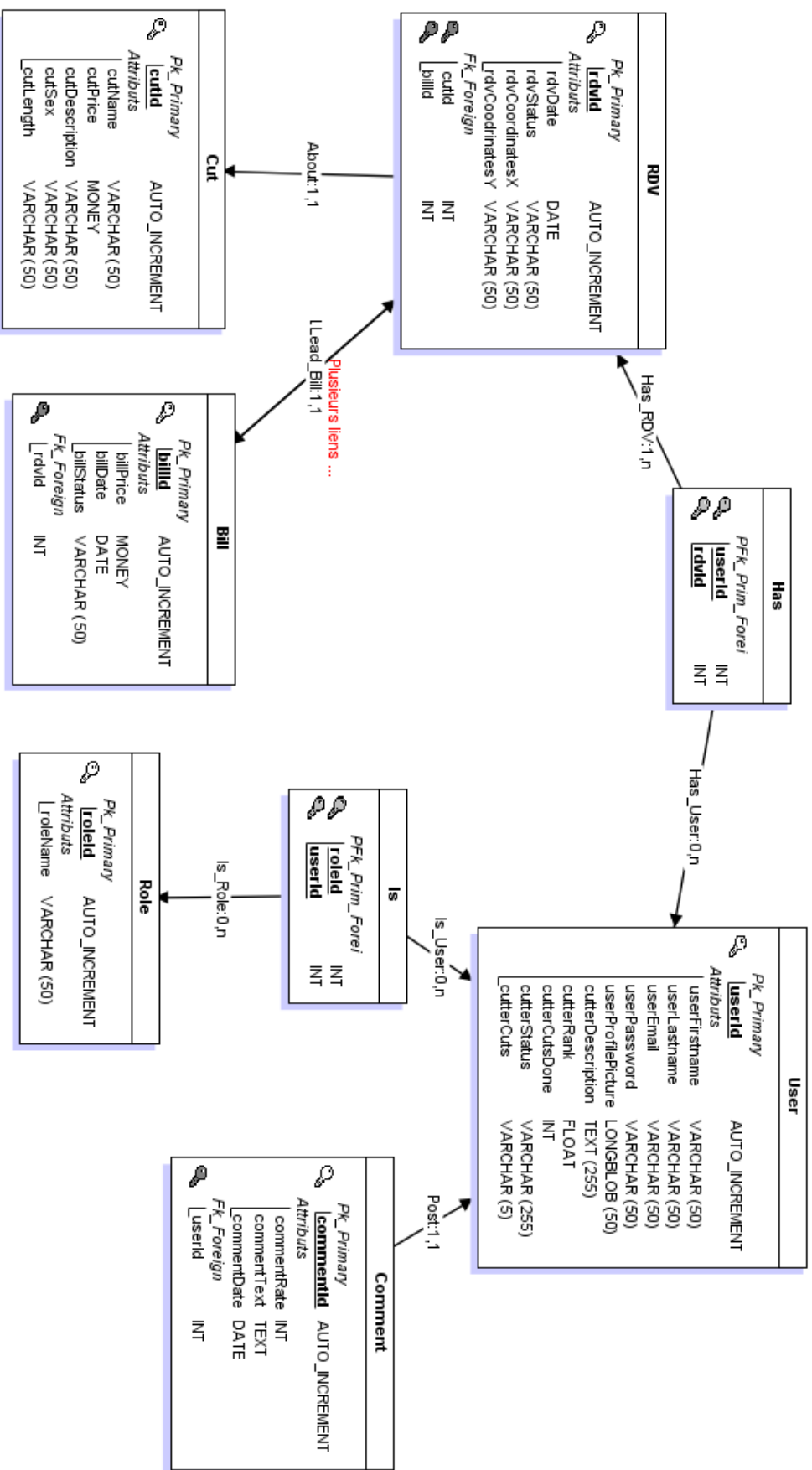


Modèle Logique de Données (MLD)

Le MLD, ou Modèle Logique de Données, est, quant à lui, une représentation structurée et formelle des données d'un système d'information.

Il s'agit d'une étape intermédiaire entre le MCD (Modèle Conceptuel de Données) et le MPD (Modèle Physique de Données).

Voici le MLD de l'application Uber Cut :



Conception de l'application

Pour la conception de Uber Cut, je me suis appuyé sur le langage UML (Unified Modeling Language) pour représenter le fonctionnement de l'application. Le langage UML est un langage graphique standard utilisé pour modéliser et représenter visuellement les systèmes logiciels. Il permet de capturer les aspects essentiels d'un système, de sa structure à son comportement, en utilisant une notation graphique compréhensible par les développeurs, les concepteurs et les parties prenantes.

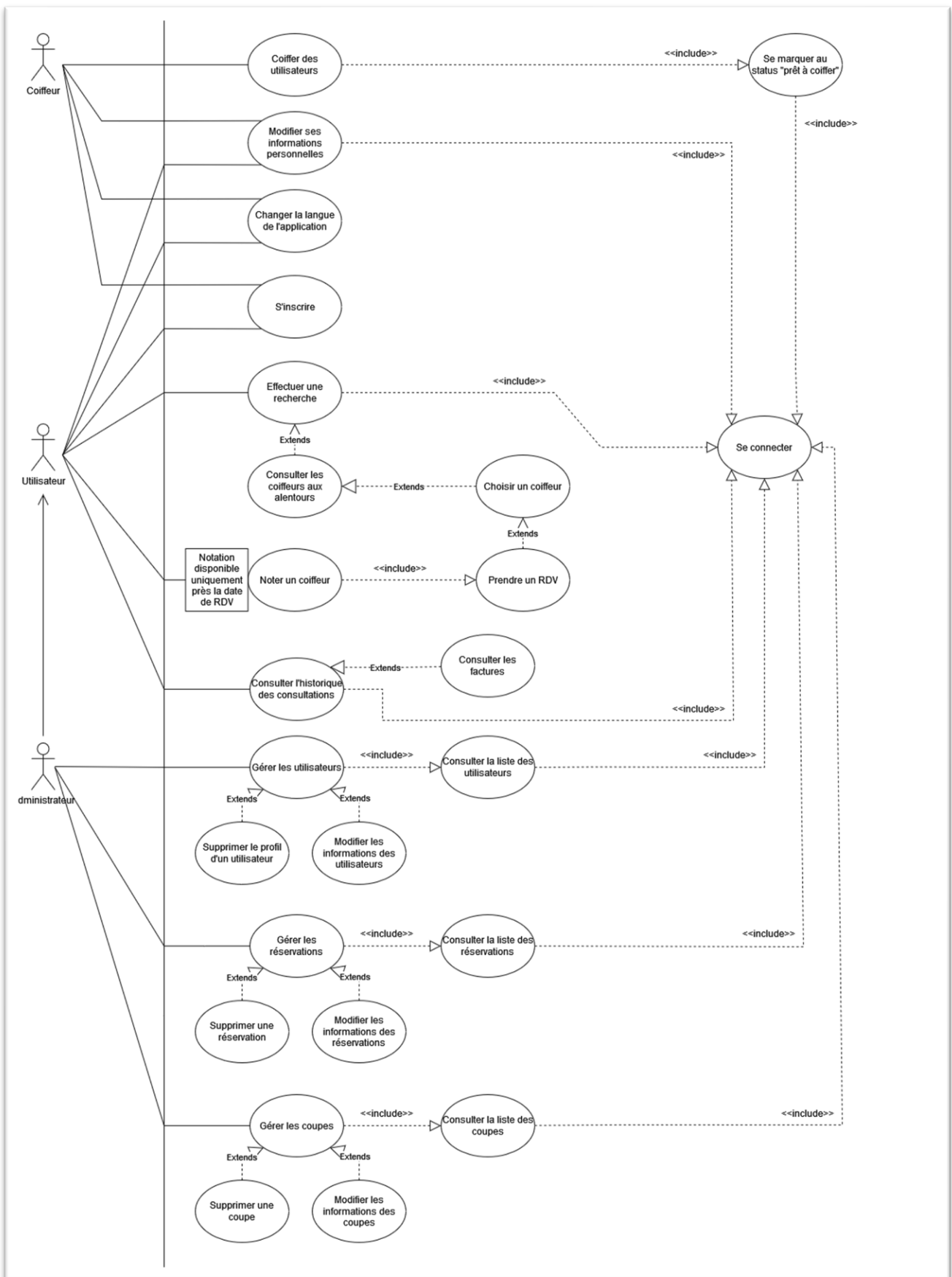
UML offre un ensemble de diagrammes qui permettent de décrire différents aspects d'un système, tels que les cas d'utilisation, les classes, les séquences, les activités, les composants, les déploiements, etc. Ces diagrammes servent de supports de communication pour analyser, concevoir, documenter et visualiser un système logiciel, facilitant ainsi la compréhension et la collaboration entre les membres de l'équipe de développement et les parties prenantes. Le langage UML est largement utilisé dans l'industrie du développement logiciel pour améliorer la modélisation, la conception et la documentation des systèmes complexes.

Parmi les schémas UML que j'ai élaborés pour Uber Cut, deux d'entre eux jouent un rôle essentiel dans la compréhension des fonctionnalités et de la structure de mon système : le diagramme de cas d'utilisation (use case) et le diagramme de classes. Ces schémas UML offrent une vision claire et structurée de mon application, permettant ainsi de visualiser les interactions entre les utilisateurs et le système, ainsi que la structure et les relations entre les différentes entités.

Diagramme de Cas d'Utilisation (Use Case)

Le diagramme de cas d'utilisation décrit les interactions entre les utilisateurs (clients, coiffeurs, administrateurs) et le système. Il identifie les différentes fonctionnalités offertes par l'application, telles que la création de compte, la réservation de séances de coiffure, la gestion de la disponibilité des coiffeurs, et l'administration de la plateforme.

Voici le diagramme de Use Case de l'application Uber Cut :



Conception multicouche MVC

Le schéma MVC (Modèle-Vue-Contrôleur) est un concept fondamental en développement logiciel, largement utilisé dans la conception d'applications web et d'autres systèmes interactifs. Il offre une structure claire et organisée pour séparer les différentes responsabilités d'une application et favoriser une meilleure maintenabilité, extensibilité et réutilisabilité du code. Voici une présentation des composants de ce schéma dans le cadre du Framework Symfony.

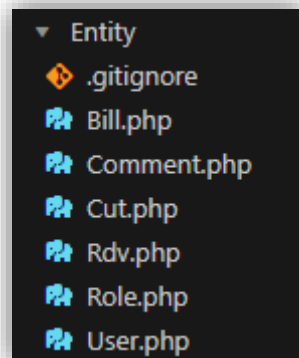
Le Modèle

Le Modèle représente la couche responsable de la gestion des données et de la logique métier. Il est chargé d'interagir avec la base de données, de récupérer et de stocker les informations pertinentes pour l'application Uber Cut.

En utilisant Symfony, j'ai pu bénéficier de l'ORM (Object Relational Mapping) Doctrine, qui facilite la création et la manipulation des entités en se basant sur les modèles de données définis. L'ORM, ici Doctrine, sert d'intermédiaire entre les classes PHP et les tables de la base de données SQL.

Le Modèle dans Symfony se compose donc de mes entités et de leurs relations, ainsi que des classes de gestion des requêtes et des opérations de validation et de persistance des données. Grâce à la clarté et à la flexibilité du Modèle dans Symfony, j'ai pu organiser mes données de manière cohérente et les manipuler de manière efficace, en garantissant la stabilité et l'intégrité de mon application.

Voici l'organisation des Entités dans le projet.



Exemple avec l'entité User Cette entité est divisée en 3 parties, la déclaration des variables de l'entité, les getters et les setters :

```
User.php x
src > Entity > User.php > User > getId
1  <?php
2
3  namespace App\Entity;
4
5  use App\Repository\UserRepository;
6  use Doctrine\ORM\Mapping as ORM;
7  use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
8  use Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;
9  use Symfony\Component\Security\Core\User\UserInterface;
10
11  #[ORM\Entity(repositoryClass: UserRepository::class)]
12  #[ORM\UniqueConstraint(name: 'UNIQ_IDENTIFIER_EMAIL', fields: ['email'])]
13  #[UniqueEntity(fields: ['email'], message: 'There is already an account with this email')]
14  class User implements UserInterface, PasswordAuthenticatedUserInterface
15  {
16      #[ORM\Id]
17      #[ORM\GeneratedValue]
18      #[ORM\Column]
19      private ?int $id = null;
20
21      #[ORM\Column(length: 180)]
22      private ?string $email = null;
23
24      #[ORM\Column(length: 255)]
25      private ?string $firstName = null;
26
27      #[ORM\Column(length: 255)]
28      private ?string $lastName = null;
29
30      #[ORM\Column(length: 20)]
31      private ?string $phoneNumber = null;
32
33      #[ORM\Column(type: 'text', nullable: true)]
34      private ?string $cutterDescription = null;
35
36      #[ORM\Column(type: 'float', nullable: true)]
37      private ?float $cutterRank = null;
38
39      #[ORM\Column(nullable: true)]
40      private ?string $profilePicture = null;
41
42      #[ORM\Column(length: 255, nullable: true)]
43      private ?string $cutterStatus = null;
44
```

Exemples de getters et setters de l'entité User :

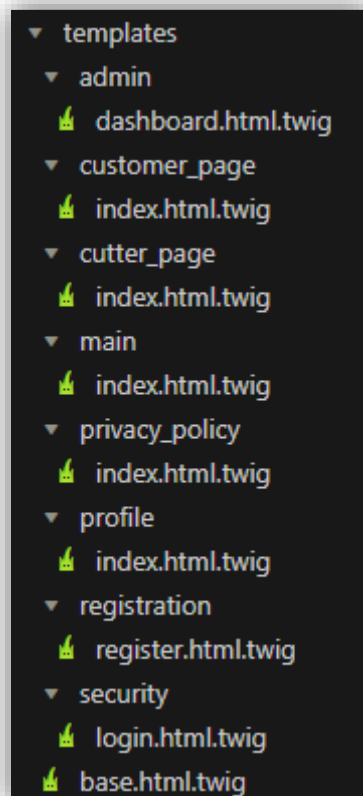
```
User.php X
src > Entity > User.php > User > setCutterDescription
14  class User implements UserInterface, PasswordAuthenticatedUserInterface
68      public function getEmail(): ?string
69      {
70          return $this->email;
71      }
72
73      public function setEmail(string $email): static
74      {
75          $this->email = $email;
76
77          return $this;
78      }
79
80      public function getFirstName(): ?string
81      {
82          return $this->firstName;
83      }
84
85      public function setFirstName(string $firstName): static
86      {
87          $this->firstName = $firstName;
88
89          return $this;
90      }
91
92      public function getLastName(): ?string
93      {
94          return $this->lastName;
95      }
96
97      public function setLastName(string $lastName): static
98      {
99          $this->lastName = $lastName;
100
101          return $this;
102      }
```

La Vue

La Vue est responsable de la présentation des informations aux utilisateurs et de l'interaction avec eux. Grâce au moteur de templates Twig fourni par Symfony, j'ai pu concevoir des vues dynamiques en utilisant des modèles réutilisables et des composants.

Les vues dans Symfony sont construites à partir des données fournies par le Contrôleur et sont responsables de l'affichage des résultats de manière structurée et conviviale. J'ai pu créer des interfaces utilisateur esthétiques et réactives en utilisant les fonctionnalités de Twig telles que les boucles, les conditions et les filtres. De plus, Symfony offre des outils pratiques pour gérer les formulaires et les interactions utilisateur, ce qui facilite la collecte de données et les actions de l'utilisateur au sein de l'application.

Voici l'organisation des fichiers Twig :



Pour cet exemple, voici le fichier Twig responsable de l'affichage de la page login (connexion) :

```

login.html.twig X
templates > security > login.html.twig
1  {% extends 'base.html.twig' %}
2
3  {% block content %}
4
5      <div class="container">
6          <div class="row justify-content-center align-items-center min-vh-100">
7              <div class="col-12 col-md-6 col-lg-4">
8                  <div class="card shadow-lg">
9                      <div class="card-body">
10                         <h1 class="h3 mb-3 font-weight-normal text-center">Connexion</h1>
11                         <form method="post">
12                             {% if error %}
13                                 <div class="alert alert-danger">{{ error.messageKey|trans(error.messageData, 'security') }}</div>
14                             {% endif %}
15
16                             {% if app.user %}
17                                 <div class="mb-3">
18                                     Vous êtes connecté en tant que {{ app.user.userIdentifier }}, <a href="{{ path('app_logout') }}">Déconnexion</a>
19                                 </div>
20                             {% endif %}
21
22                             <div class="form-group">
23                                 <label for="inputEmail">Email</label>
24                                 <input type="email" value="{{ last_username }}" name="email" id="inputEmail" class="form-control" autocomplete="email" required autofocus>
25                             </div>
26
27                             <div class="form-group">
28                                 <label for="inputPassword">Mot de passe</label>
29                                 <input type="password" name="password" id="inputPassword" class="form-control" autocomplete="current-password" required>
30                             </div>
31
32                             <input type="hidden" name="_csrf_token" value="{{ csrf_token('authenticate') }}">
33
34                             <div class="text-center">
35                                 <a href="{{ path('app_register') }}" class="btn btn-link">Vous n'avez pas de compte ?</a>
36                                 <button class="btn btn-primary btn-block" type="submit">Se connecter</button>
37                             </div>
38                         </form>
39                     </div>
40                 </div>
41             </div>
42         </div>
43     </div>
44
45 {% endblock %}

```

Le code présenté est un template Twig pour une page de connexion (login.html.twig) dans une application Symfony. Il étend un template de base (base.html.twig) et utilise les fonctionnalités de Twig pour créer une interface utilisateur pour la connexion.

Dans ce cas précis, l'utilisation du `{% extends 'base.html.twig' %}` permet d'hériter de la barre de navigation ainsi qu'à toutes les librairies CSS et JS permettant le bon fonctionnement de toutes les pages de l'application.

Le Contrôleur

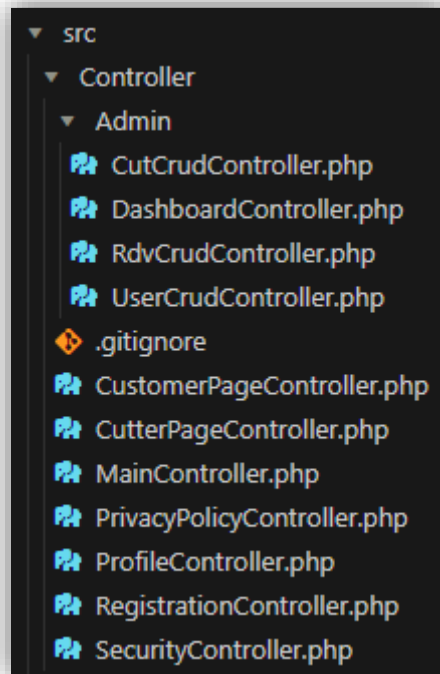
Le Contrôleur joue un rôle crucial dans la gestion des actions de l'utilisateur et l'orchestration des interactions entre le Modèle et la Vue. Dans mon projet Symfony, le Contrôleur agit comme un intermédiaire entre les requêtes entrantes et les opérations à effectuer.

Le rôle du contrôleur est de recevoir les données saisies par l'utilisateur sous forme de requête et les transmet au Modèle pour effectuer des opérations telles que la création, la modification ou la suppression de données. Une fois que le Modèle a effectué les opérations nécessaires, le Contrôleur met à jour la Vue pour refléter les changements et renvoie la réponse appropriée à l'utilisateur.

Symfony fournit un système de routage qui associe les URL aux actions des Contrôleurs, facilitant ainsi la gestion des différentes fonctionnalités de l'application.

Chaque action d'un contrôleur correspond à une méthode qui traite une requête spécifique.

Voici l'organisation des contrôleurs de l'application :



Pour cet exemple, voici le contrôleur de la page Dashboard destiné aux coiffeurs :

Il est important de noter que les erreurs rouges du code ne sont que des bugs de lecture de mon IDE, la déclaration ligne 5 ainsi que les utilisations de la fonction `getCutterStatus` & `setCutterStatus` fonctionnent bien.

```

CutterPageController.php 4, M
src > Controller > CutterPageController.php > ...
1  <?php
2
3  namespace App\Controller;
4
5  use App\Entity\User;
6  use Doctrine\ORM\EntityManagerInterface;
7  use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
8  use Symfony\Component\HttpFoundation\Request;
9  use Symfony\Component\HttpFoundation\Response;
10 use Symfony\Component\Routing\Annotation\Route;
11 use Symfony\Component\Security\Core\User\UserInterface;
12
13 class CutterPageController extends AbstractController
14 {
15     #[Route('/cutter', name: 'app_cutter_page')]
16     public function index(): Response
17     {
18         $user = $this->getUser();
19
20         return $this->render('cutter_page/index.html.twig', [
21             'controller_name' => 'CutterPageController',
22             'user' => $user,
23             'cutter_status' => $user->getCutterStatus(),
24         ]);
25     }
26
27     #[Route('/update-cutter-description', name: 'update_cutter_description', methods: ['POST'])]
28     public function updateCutterDescription(Request $request, EntityManagerInterface $em, UserInterface $user): Response
29     {
30         $description = $request->request->get('cutterdescription');
31         if ($description !== null) {
32             $user->setCutterDescription($description);
33             $em->persist($user);
34             $em->flush();
35         }
36
37         return $this->redirectToRoute('app_cutter_page');
38     }
39
40     #[Route('/update-cutter-status', name: 'update_cutter_status', methods: ['POST'])]
41     public function updateCutterStatus(EntityManagerInterface $em, UserInterface $user): Response
42     {
43         $currentStatus = $user->getCutterStatus();
44         $newStatus = $currentStatus === 'LFC' ? 'IDLE' : 'LFC';
45         $user->setCutterStatus($newStatus);
46         $em->persist($user);
47         $em->flush();
48
49         return new Response('Status updated to ' . $newStatus);
50     }
51 }

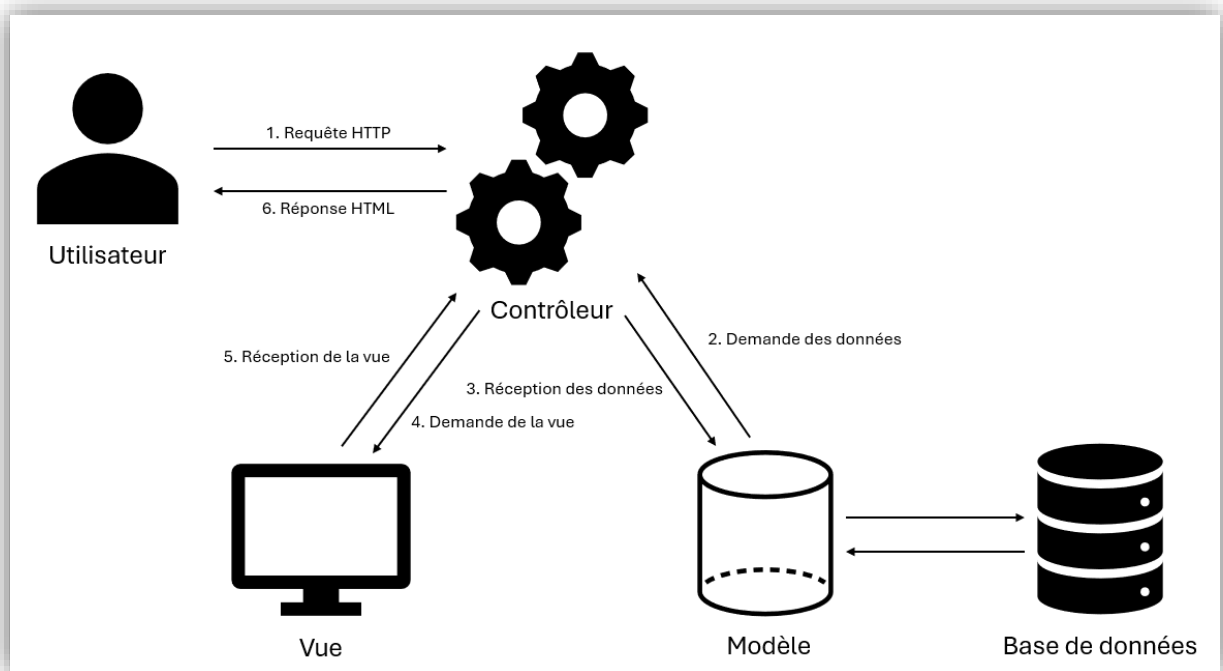
```

Communication entre les 3 composants

Chaque composant de l'application a sa fonction, nous allons nous pencher sur les échanges de données opérés afin de fournir à l'utilisateur des réponses à ses requêtes.

1. L'utilisateur navigue dans l'application par le biais de requêtes http dans son URL, chaque URL est liée à une Route qui redirige l'application vers le mon contrôleur.
C'est le contrôleur qui déterminera ensuite les fonctionnalités de la page web.

2. Afin d'afficher les informations correctes à l'utilisateur, le contrôleur fait une requête au Modèle en passant par l'ORM de Doctrine.
C'est primordial dans le cas où il faudrait afficher le nom de l'utilisateur de la session actuelle par exemple.
3. Le Modèle renvoie les données requêtées au contrôleur.
4. Le Contrôleur détermine la Vue (Twig) utilisée pour l'affichage de la réponse.
5. Une fois le template Twig et les données récupérées, le contrôleur se charge de mettre en forme les données dans la vue.
6. La réponse http est prête, elle peut donc être affichée à l'utilisateur.



Dans le cadre de la page de recherche de coiffeur disponible dans l'application Uber Cut, l'échange d'informations entre les différents éléments du framework est particulièrement intéressant.

Les coiffeurs correspondant à la recherche de l'utilisateur sont affichés sous forme d'épingles sur la carte.

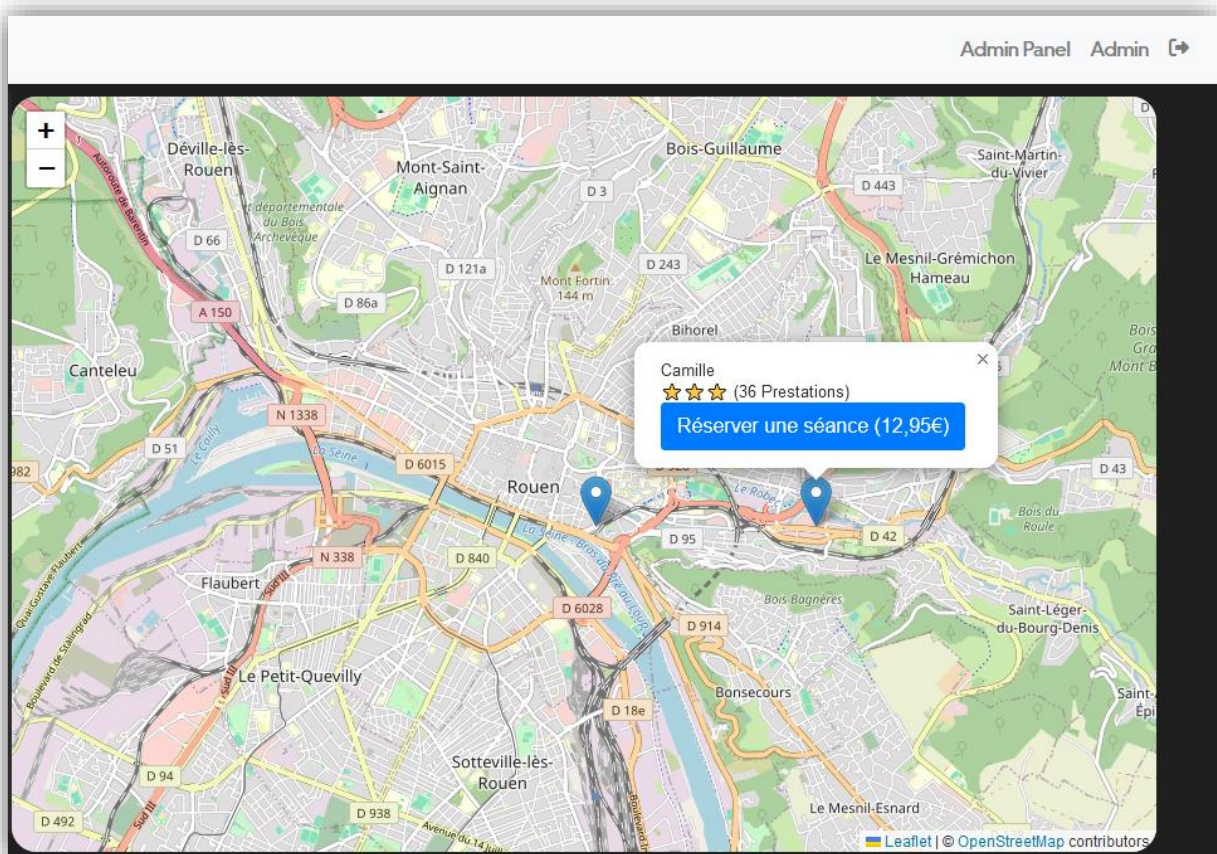
Lorsqu'on clique sur une de ces épingles, le profil du coiffeur apparaît avec la possibilité de demander une séance avec celui-ci.

L'affichage de la carte se fait par le biais de l'API Leaflet sur la partie Front-End de l'application en langage JavaScript.

Les informations nécessaires à l'affichage du profil d'un coiffeur passent donc par plusieurs étapes :

- 1- **Requête de l'utilisateur** : L'utilisateur effectue une recherche en saisissant des critères spécifiques (comme la localisation, le type de service, etc.) dans l'interface utilisateur.

- 2- **Le contrôleur fait une requête à l'entité** : Le contrôleur (ou le routeur) de l'application capture cette requête et la transmet à l'entité appropriée. L'entité représente les modèles de données de l'application.
- 3- **L'entité récupère les informations de la base de données** : L'entité interagit avec la base de données pour récupérer les informations des coiffeurs qui correspondent aux critères de recherche de l'utilisateur.
- 4- **Le contrôleur envoie les informations d'affichage du profil coiffeur à la Vue Twig sous forme de JSON** : Une fois les données récupérées, le contrôleur les transforme en un format JSON approprié et les envoie à la Vue Twig. La Vue Twig est responsable de générer le code HTML qui sera envoyé au navigateur de l'utilisateur.
- 5- **Le code JavaScript traite les données afin de les afficher correctement sur la carte** : Le code JavaScript s'exécute côté client, utilisant l'API Leaflet pour afficher les épingles des coiffeurs sur la carte. Lorsque l'utilisateur clique sur une épingle, le code JavaScript récupère les données du profil du coiffeur depuis le JSON et affiche les informations détaillées du coiffeur dans un pop-up ou une section dédiée de l'interface utilisateur.



L'architecture 3 tiers

L'architecture 3 tiers est un modèle d'organisation très efficace pour les applications modernes. Elle permet de structurer l'application de manière modulaire, ce qui facilite la maintenance, l'évolutivité et la répartition des responsabilités.

Voici une description détaillée de chaque couche, ainsi qu'un schéma de l'architecture 3 tiers appliqué à une application utilisant Hostinger pour le déploiement et la gestion des bases de données.

La première couche, la présentation ou couche d'interface utilisateur, est chargée de l'interaction avec les utilisateurs finaux. Elle comprend l'interface graphique, les contrôles et les éléments visuels qui permettent aux utilisateurs d'interagir avec l'application. Cette couche est souvent implémentée en utilisant le pattern MVC, où le contrôleur gère les interactions utilisateur et coordonne les actions à effectuer.

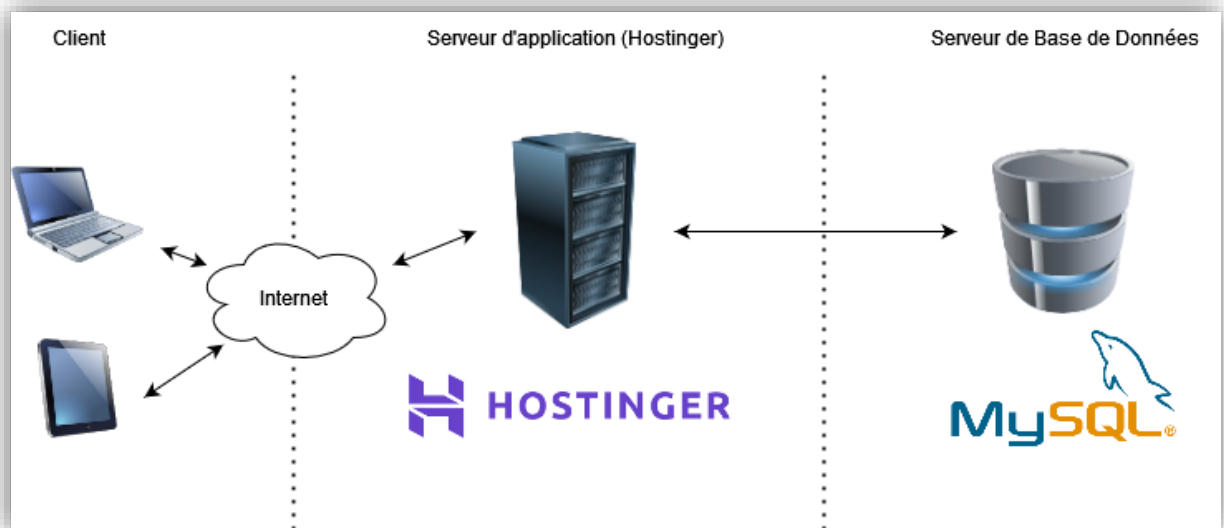
La deuxième couche, la logique métier ou couche de traitement, contient la logique métier de l'application. C'est ici que les règles métier, les calculs complexes et les opérations spécifiques à l'application sont mises en œuvre. Cette couche utilise les données provenant de la couche de données et les transforme en résultats significatifs pour les utilisateurs.

Enfin, la troisième couche, la couche d'accès aux données, est responsable de l'accès et de la gestion des données de l'application. Elle peut être liée à une base de données, à des services web ou à d'autres sources de données. Cette couche fournit les mécanismes nécessaires pour récupérer, stocker et manipuler les données de manière sécurisée.

En adoptant une architecture trois tiers, nous pouvons bénéficier d'une meilleure modularité, d'une séparation claire des responsabilités et d'une évolutivité accrue. Chaque couche peut être développée et testée indépendamment, ce qui facilite la maintenance et les mises à jour ultérieures de l'application.

En intégrant cette architecture, Uber Cut peut non seulement améliorer l'efficacité et la réactivité de son service, mais aussi faciliter l'introduction de nouvelles fonctionnalités et le maintien d'une performance optimale. Cette approche modulaire assure que chaque partie de l'application peut évoluer indépendamment, répondant aux besoins changeants des utilisateurs et du marché.

Voici le schéma représentant l'architecture 3 tiers :



Sécurité

Dans le paysage numérique actuel, la sécurité des applications web est devenue une préoccupation majeure. Les applications web jouent un rôle essentiel dans la communication, les transactions en ligne et le stockage des données sensibles. Cependant, elles sont également vulnérables aux attaques malveillantes et aux violations de la confidentialité. Les cybercriminels utilisent des techniques sophistiquées pour exploiter les failles de sécurité des applications web et accéder aux données sensibles des utilisateurs. Les conséquences peuvent être désastreuses pour les entreprises et les utilisateurs, allant de la perte de données à la violation de la vie privée.

La protection des utilisateurs, la préservation de l'intégrité des données et la disponibilité des services sont des aspects essentiels pour assurer la confiance et la fiabilité d'une application web. C'est pourquoi la sécurité est un enjeu crucial à prendre en compte lors du développement d'une application web. Les développeurs doivent être conscients des risques liés à la sécurité et mettre en place des mesures de sécurité efficaces pour protéger les utilisateurs et les données sensibles.

Les attaques XSS (Cross-Script Scripting)

Les attaques XSS sont l'une des menaces les plus courantes pour les applications web. Elles se produisent lorsque des attaquants injectent des scripts malveillants dans des pages web consultées par d'autres utilisateurs. Ces scripts peuvent être utilisés pour voler des cookies, des sessions ou d'autres informations sensibles, ou pour rediriger les utilisateurs vers des sites web malveillants.

Le framework Symfony protège par défaut ses formulaires d'éventuelles attaques XSS.

Voici un exemple d'utilisation de Twig pour échapper les données dans un formulaire d'inscription :

```
register.html.twig
templates > registration > register.html.twig
1 {% extends 'base.html.twig' %}
2
3 {% block content %}
4     <div class="container">
5         <div class="row justify-content-center align-items-center min-vh-100">
6             <div class="col-12 col-md-8 col-lg-6">
7                 <div class="card shadow-lg" style="margin-top:20px">
8                     <div class="card-body p-4">
9                         <h1 class="h3 mb-3 font-weight-normal text-center">Inscription</h1>
10
11                         {{ form_errors(registrationForm) }}
12
13                         {{ form_start(registrationForm) }}
14                             <div class="form-group">
15                                 {{ form_row(registrationForm.email, {'attr': {'class': 'form-control'}, 'label': 'Email'}) }}
16                             </div>
17                             <div class="form-group">
18                                 {{ form_row(registrationForm.firstName, {'attr': {'class': 'form-control'}, 'label': 'Prénom'}) }}
19                             </div>
20                             <div class="form-group">
21                                 {{ form_row(registrationForm.lastName, {'attr': {'class': 'form-control'}, 'label': 'Nom'}) }}
22                             </div>
23                             <div class="form-group">
24                                 {{ form_row(registrationForm.phoneNumber, {'attr': {'class': 'form-control'}, 'label': 'Numéro de téléphone'}) }}
25                             </div>
26                             <div class="form-group">
27                                 {{ form_row(registrationForm.plainPassword, {
28                                     'label': 'Mot de passe', 'attr': {'class': 'form-control'}
29                                 }) }}
30                             </div>
31                             <div class="form-group">
32                                 {{ form_row(registrationForm.confirmPassword, {
33                                     'label': 'Confirmez le mot de passe', 'attr': {'class': 'form-control'}
34                                 }) }}
35                             </div>
36                             <div class="form-group form-check">
37                                 <a href="{{ path('app_login') }}" class="btn btn-link">Conditions générales d'utilisation</a>
38                                 {{ form_row(registrationForm.agreeTerms, {'label': 'Accepter les conditions'}) }}
39                             </div>
40                             <div class="text-center mb-3">
41                                 <a href="{{ path('app_login') }}" class="btn btn-link">Vous avez déjà un compte ?</a>
42                             </div>
43                             <div class="text-center mb-3">
44                                 <button type="submit" class="btn btn-primary btn-block">Inscription</button>
45                             </div>
46                         </div>
47                     </div>
48                 </div>
49             </div>
50         </div>
51     </div>
52 {% endblock %}
```

Dans cet exemple, toutes les données provenant du formulaire sont filtrées par défaut par Twig, ce qui empêche l'injection de scripts malveillants.

Les injections SQL

Les injections SQL se produisent lorsque des entrées non validées sont insérées directement dans une requête SQL, permettant à un attaquant de manipuler la base de données. Cela peut entraîner l'accès non autorisé à des données, la corruption de données ou la divulgation d'informations sensibles.

Symfony utilise Doctrine ORM pour interagir avec les bases de données de manière sécurisée. Voici un exemple d'utilisation de requêtes paramétrées avec Doctrine dans `UserRepository` :

```

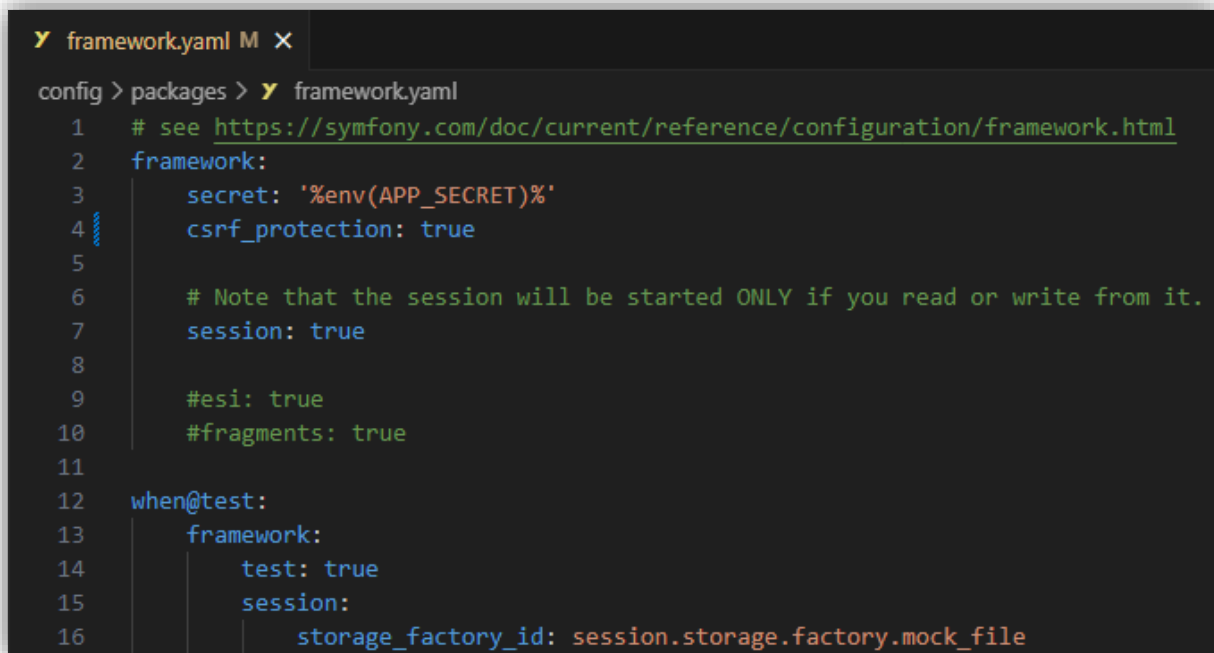
UserRepository.php X
src > Repository > UserRepository.php > ...
1  <?php
2
3  namespace App\Repository;
4
5  use App\Entity\User;
6  use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
7  use Doctrine\Persistence\ManagerRegistry;
8  use Symfony\Component\Security\Core\Exception\UnsupportedUserException;
9  use Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;
10 use Symfony\Component\Security\Core\User\PasswordUpgraderInterface;
11
12 /**
13  * @extends ServiceEntityRepository<User>
14  */
15 class UserRepository extends ServiceEntityRepository implements PasswordUpgraderInterface
16 {
17     public function __construct(ManagerRegistry $registry)
18     {
19         parent::__construct($registry, User::class);
20     }
21
22     /**
23      * Used to upgrade (rehash) the user's password automatically over time.
24      */
25     public function upgradePassword(PasswordAuthenticatedUserInterface $user, string $newHashedPassword): void
26     {
27         if (!$user instanceof User) {
28             throw new UnsupportedUserException(sprintf('Instances of "%s" are not supported.', $user::class));
29         }
30
31         $user->setPassword($newHashedPassword);
32         $this->getEntityManager()->persist($user);
33         $this->getEntityManager()->flush();
34     }

```

Les attaques CSRF (Cross-Site Request Forgery)

Les attaques CSRF exploitent la confiance qu'un site web a envers le navigateur d'un utilisateur. Un attaquant peut amener un utilisateur authentifié à effectuer une action non désirée sur un site web où il est connecté, sans que l'utilisateur ne s'en aperçoive. Cela peut conduire à des modifications de données, des transactions non autorisées ou d'autres actions malveillantes.

Symfony offre une protection intégrée contre les attaques CSRF. Pour activer cette protection, vous devez configurer le fichier `framework.yaml` comme suit :



```
Y framework.yaml M X
config > packages > Y framework.yaml
1 # see https://symfony.com/doc/current/reference/configuration/framework.html
2 framework:
3     secret: '%env(APP_SECRET)%'
4     csrf_protection: true
5
6     # Note that the session will be started ONLY if you read or write from it.
7     session: true
8
9     #esi: true
10    #fragments: true
11
12 when@test:
13     framework:
14         test: true
15         session:
16             storage_factory_id: session.storage.factory.mock_file
```

Avec `csrf_protection` activé, Symfony génère et vérifie automatiquement les tokens CSRF, protégeant ainsi les formulaires contre les attaques CSRF.

Les tests unitaires

Les tests unitaires sont une méthode essentielle dans le processus de développement logiciel. Ils permettent de vérifier le bon fonctionnement de composants individuels d'une application, appelés "unités" ou "modules". Chaque unité est testée isolément pour s'assurer qu'elle fonctionne correctement indépendamment des autres parties du système.

Cette approche joue un rôle crucial en permettant la détection précoce des erreurs, facilitant ainsi leur correction avant qu'elles ne se propagent dans le système. En outre, les tests unitaires permettent aux développeurs de modifier le code en toute confiance, sachant que les tests fourniront une garantie contre l'introduction de nouvelles erreurs. Ils servent également de documentation vivante du comportement attendu de chaque unité, montrant comment chaque partie du code est censée fonctionner et réagir dans différentes situations. De plus, une couverture de tests élevée renforce la robustesse du code, réduisant les risques de bugs cachés.

Dans ce chapitre, nous nous concentrerons sur les tests unitaires appliqués à l'entité `User` d'une application Symfony. L'entité `User` représente un utilisateur du système avec divers attributs tels que l'email, le prénom, le nom, le numéro de téléphone, la

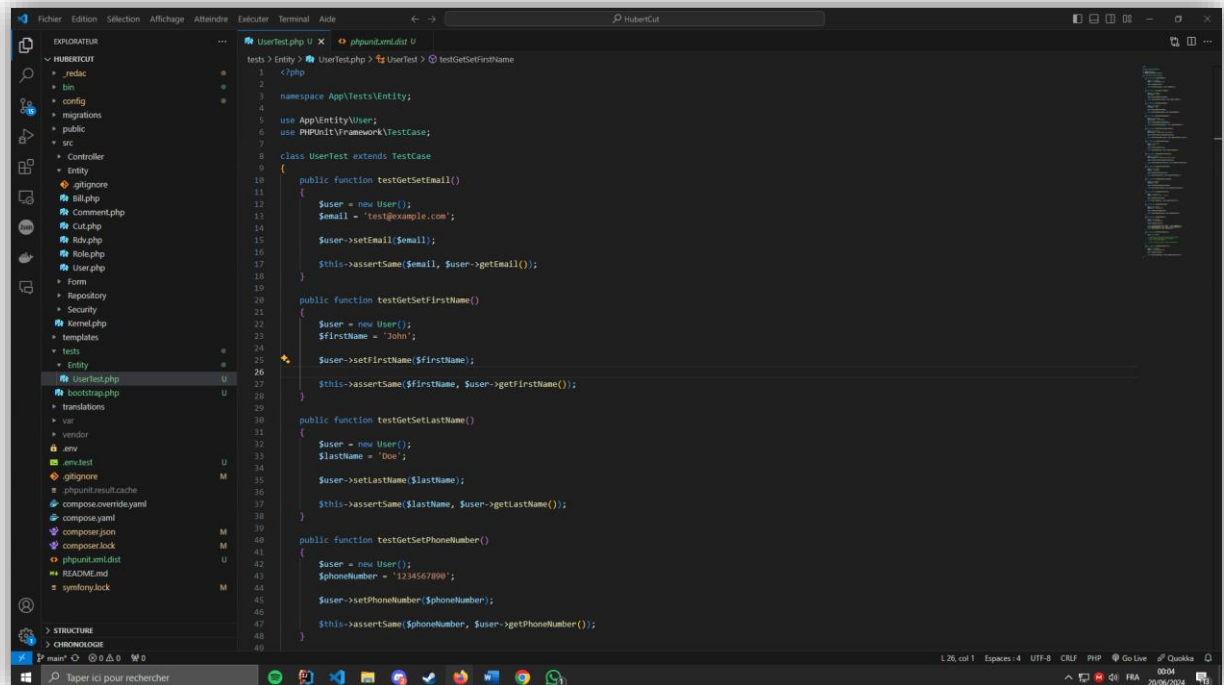
description, le rang, la photo de profil, le statut, le nombre de coupes réalisées et les rôles.

J'ai donc créé un set de tests unitaires simulant des valeurs pour chaque fonction de l'entité User.

```
UserTest.php U X  phpunit.xml.dist U
tests > Entity > UserTest.php > UserTest > testGetSetPhoneNumber
1  <?php
2
3  namespace App\Tests\Entity;
4
5  use App\Entity\User;
6  use PHPUnit\Framework\TestCase;
7
8  class UserTest extends TestCase
9  {
10     public function testGetSetEmail()
11     {
12         $user = new User();
13         $email = 'test@example.com';
14
15         $user->setEmail($email);
16
17         $this->assertSame($email, $user->getEmail());
18     }
19
20     public function testGetSetFirstName()
21     {
22         $user = new User();
23         $firstName = 'John';
24
25         $user->setFirstName($firstName);
26
27         $this->assertSame($firstName, $user->getFirstName());
28     }
29
30     public function testGetSetLastName()
31     {
32         $user = new User();
33         $lastName = 'Doe';
34
35         $user->setLastName($lastName);
36
37         $this->assertSame($lastName, $user->getLastName());
38     }
39 }
```


A l'exécution des tests unitaires, j'aurais donc un retour sur chaque test avec une réussite ou un échec.

Exemple :



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'config', 'migrations', 'public', 'src', 'tests', and 'vendor'. The code editor displays a PHP file named 'UserTest.php' with the following content:

```
1 <?php
2
3 namespace App\Tests\Entity;
4
5 use App\Entity\User;
6 use PHPUnit\Framework\TestCase;
7
8 class UserTest extends TestCase
9 {
10     public function testGetSetEmail()
11     {
12         $user = new User();
13         $email = 'test@example.com';
14
15         $user->setEmail($email);
16
17         $this->assertSame($email, $user->getEmail());
18     }
19
20     public function testGetSetFirstName()
21     {
22         $user = new User();
23         $firstName = 'John';
24
25         $user->setFirstName($firstName);
26
27         $this->assertSame($firstName, $user->getFirstName());
28     }
29
30     public function testGetSetLastName()
31     {
32         $user = new User();
33         $lastName = 'Doe';
34
35         $user->setLastName($lastName);
36
37         $this->assertSame($lastName, $user->getLastName());
38     }
39
40     public function testGetSetPhoneNumber()
41     {
42         $user = new User();
43         $phoneNumber = '1234567890';
44
45         $user->setPhoneNumber($phoneNumber);
46
47         $this->assertSame($phoneNumber, $user->getPhoneNumber());
48     }
49 }
```

Veille Technologique

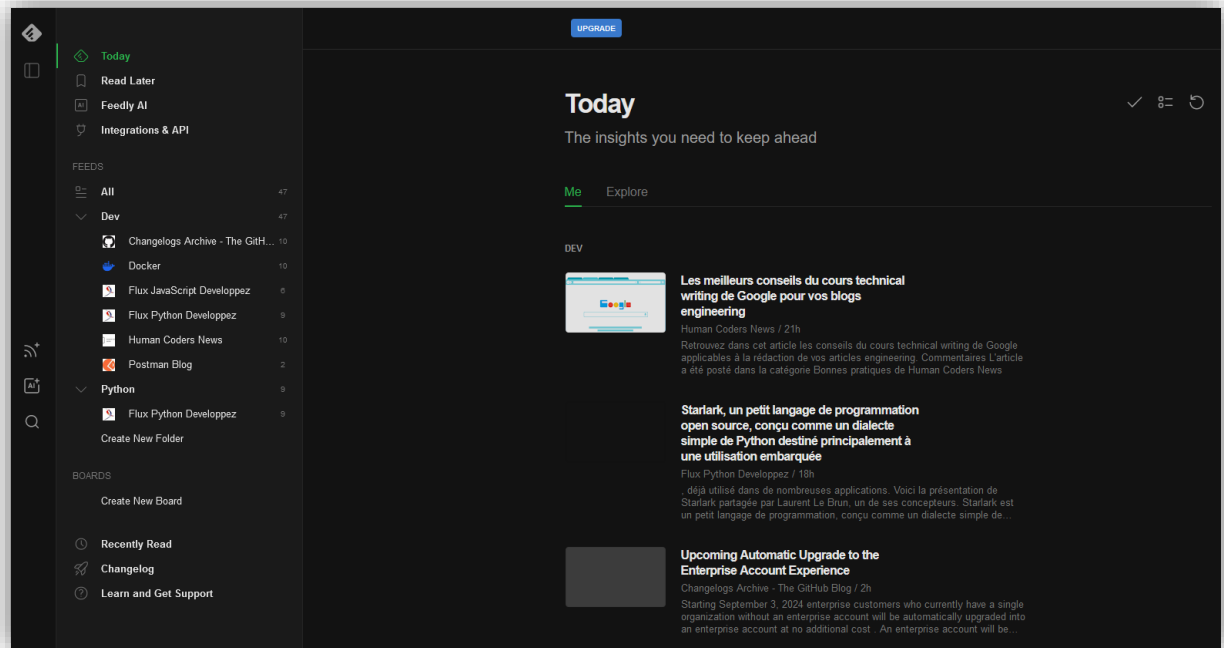
Veille globale

La veille informatique est une pratique essentielle pour rester informé des dernières évolutions technologiques et des bonnes pratiques dans son domaine. Pour faire de la veille informatique, il est crucial d'identifier ses objectifs, ses besoins, ses sources et ses outils de veille. On peut utiliser des flux RSS, Twitter, Google Alert, des newsletters, des conférences ou des discussions avec ses pairs.

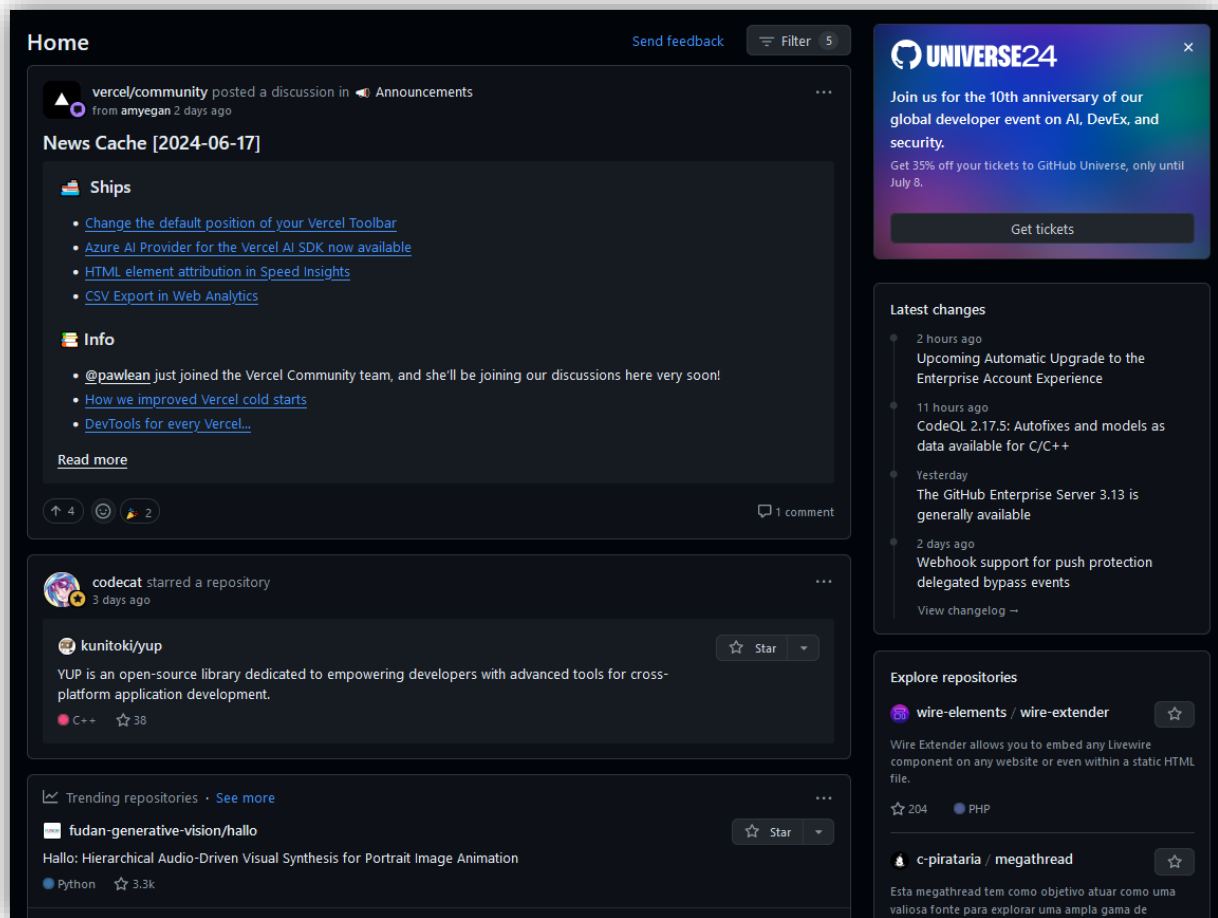
Pour ma part, étant passionné par le développement fullstack et constamment à l'affût des dernières tendances et innovations, je suis convaincu que la veille technologique est indispensable pour rester compétitif dans ce secteur en constante évolution.

Je crois que la curation de contenu est un élément clé de la veille technologique. Sélectionner les sources d'information les plus pertinentes et filtrer les informations pour ne retenir que celles qui sont les plus utiles est essentiel.

Pour cela, j'utilise des sites tels que [Feedly](#) pour suivre les dernières actualités en développement fullstack. Feedly me permet de regrouper en un seul endroit les informations provenant de différentes sources fiables et de les consulter de manière efficace.



En plus de Feedly, je suis abonné à plusieurs projets et développeurs sur GitHub. Cela remplit mon fil de nouveautés à chaque connexion, me tenant informé des dernières mises à jour de projets open-source et des nouvelles contributions de développeurs influents. Suivre ces projets et développeurs sur GitHub m'offre une perspective directe sur les tendances actuelles et les innovations dans le développement web.



La veille technologique ne se limite pas seulement à la lecture d'articles et de blogs. Participer à des conférences au sein de mon campus et discuter avec des pairs sont également des pratiques importantes. Ces interactions directes permettent d'échanger des idées, de poser des questions spécifiques et de recevoir des réponses concrètes, souvent plus enrichissantes que la simple lecture de contenu.

En somme, la veille technologique est une pratique indispensable pour tout développeur souhaitant rester à jour dans son domaine et continuer à progresser professionnellement. Elle permet de rester informé des dernières innovations, d'améliorer ses compétences et d'anticiper les futures évolutions du secteur.

Veille sécurité

Parmi les différents aspects de la veille technologique, la sécurité est indéniablement un enjeu majeur. En effet, elle concerne la protection des données, des systèmes et des utilisateurs face aux menaces et aux risques liés au cyberspace.

Pour réaliser une veille efficace sur la sécurité, il est nécessaire de consulter des sources fiables et reconnues, qui fournissent des informations pertinentes, actualisées et adaptées aux besoins des professionnels. Parmi ces sources, deux sites se distinguent par leur qualité et leur notoriété : le site de l'Agence nationale de la sécurité des systèmes d'information (ANSSI) et le site de l'Open Web Application Security Project (OWASP).

Le site de l'ANSSI est le portail officiel de la cybersécurité en France. Il propose des actualités, des guides, des recommandations, des outils et des services pour aider les acteurs publics et privés à se protéger contre les cyberattaques. Le site de l'ANSSI couvre tous les aspects de la sécurité des systèmes d'information, tels que la gouvernance, la gestion des risques, la défense en profondeur, la sensibilisation, la réaction aux incidents, etc.

L'ANSSI publie régulièrement des rapports et des alertes de sécurité, qui permettent de rester informé des nouvelles menaces et des bonnes pratiques à adopter. Ses guides de recommandations sont particulièrement précieux pour mettre en place des mesures de sécurité efficaces et conformes aux normes en vigueur.



Publications de l'ANSSI

Type de publication

Sujet de la publication

Public

Trier par entités

Thématique

Niveau de technicité

Appliquer

SÉCURISATION D'UNE
INFRASTRUCTURE VMWARE

LES FONDAMENTAUX

Publié le 10 Mai 2024 , mis à jour le 13 Mai 2024

Sécurisation d'une infrastructure VMware

Ces formats intermédiaires reflètent le point de vue de l'ANSSI au moment de leur publication et ne se positionnent pas comme des documents de recommandations détaillées, tels que les guides. Ils sont susceptibles d'être mis à jour régulièrement suivant l'évolution de la menace et des technologies utilisées, les retours d'expérience, etc.

RECOMMANDATIONS DE SÉCURITÉ
RELATIVES AU DÉPLOIEMENT D'UNE
INFRASTRUCTURE OPENSTACK POUR
UN SERVICE IAAS SECNUMCLOUD

GUIDE ANSSI

Publié le 03 Mai 2024 , mis à jour le 03 Mai 2024

Recommandations de déploiement d'un service IAAS OpenStack SecNumCloud

Ce guide est destiné à des personnes chargées de la mise en œuvre d'une infrastructure OpenStack sécurisée. Il présente des exemples de solutions permettant d'être conforme aux exigences du référentiel SecNumCloud qui ne sont pas directement couvertes par une fonctionnalité implémentée dans la suite logicielle.

RECOMMANDATIONS DE SÉCURITÉ
POUR UN SYSTÈME D'IA GÉNÉRATIVE

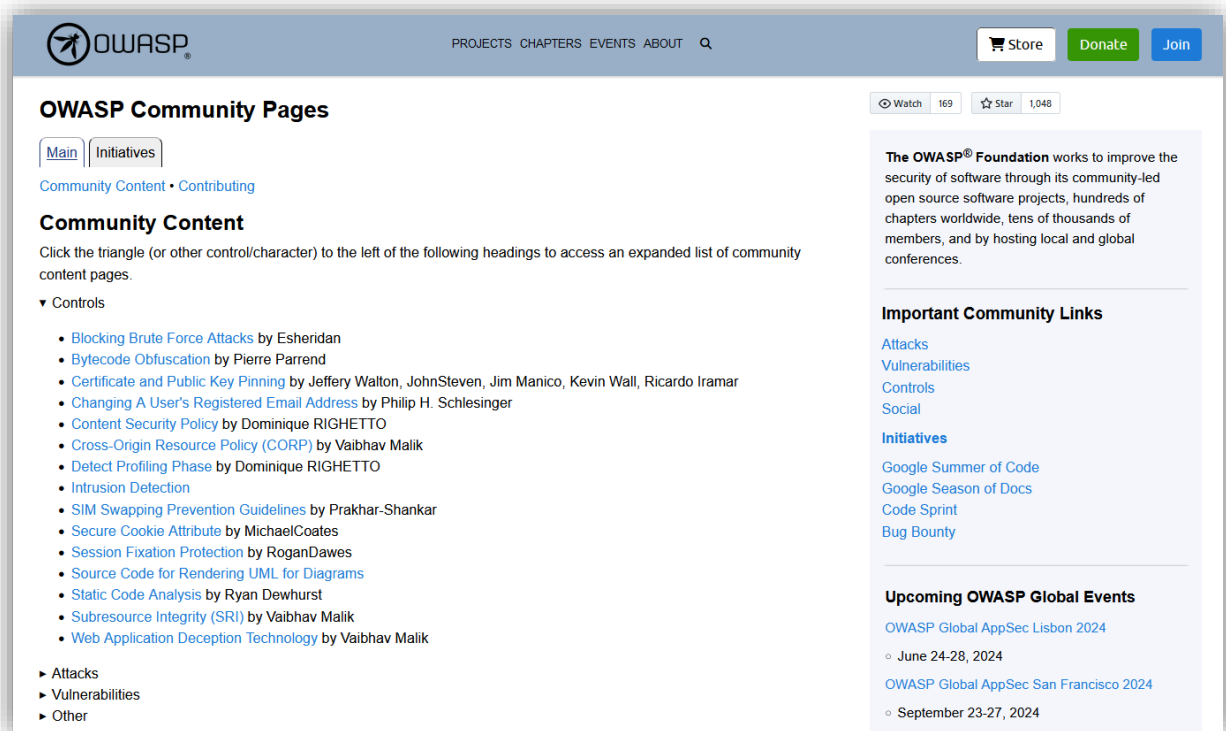
GUIDE ANSSI

Publié le 29 Avril 2024 , mis à jour le 29 Avril 2024

Recommandations de sécurité pour un système d'IA générative

Le document vise à sensibiliser les administrations et entreprises aux risques liés à l'IA générative ainsi qu'à promouvoir les bonnes pratiques dans la mise en œuvre de ce type de système.

Le site de l'OWASP est une communauté internationale qui promeut la sécurité des applications web. Il offre des ressources gratuites et ouvertes, telles que des standards, des méthodologies, des projets, des événements et des formations pour aider les développeurs, les testeurs, les auditeurs et les utilisateurs à concevoir, développer, tester et maintenir des applications web sécurisées.



Les projets OWASP, tels que le OWASP Top 10, ZAP (Zed Attack Proxy) et ASVS (Application Security Verification Standard), sont des outils et des méthodologies largement utilisés dans l'industrie pour améliorer la sécurité des applications web. En suivant les mises à jour et les nouvelles publications d'OWASP, on peut s'assurer de rester à la pointe des meilleures pratiques en matière de sécurité applicative.

En somme, pour une veille efficace sur la sécurité, il est crucial de s'appuyer sur des sources fiables et reconnues comme l'ANSSI et l'OWASP. Ces sites fournissent des informations précieuses et actualisées qui permettent de protéger les données, les systèmes et les utilisateurs contre les cybermenaces. En intégrant ces ressources dans une stratégie de veille continue, les professionnels peuvent anticiper les risques et adopter des pratiques de sécurité robustes et adaptées aux défis actuels.

Difficultés rencontrées

Lors du développement de l'application Uber Cut, j'ai rencontré plusieurs défis techniques.

L'un des plus grands obstacles a été la nécessité de revenir en arrière à plusieurs reprises en raison de problèmes de conception. La mauvaise conception initiale a souvent conduit à des révisions complètes de certaines parties du code et de la base de données, ce qui a ralenti le progrès et ajouté un niveau de complexité supplémentaire au projet.

De plus, créer une application unique sans réel équivalent existant a rendu les recherches de solutions et de code plus difficiles. Contrairement à des projets plus communs, où de nombreuses ressources et exemples sont disponibles en ligne, je me suis souvent retrouvé face à des défis pour lesquels il n'existait pas de solutions toutes prêtes. Bien que la documentation de Symfony soit très complète et que Stackoverflow soit une ressource précieuse, ces outils n'ont pas toujours suffi à résoudre certains problèmes spécifiques à Uber Cut.

Ces difficultés m'ont appris l'importance d'une planification minutieuse et d'une conception rigoureuse dès le départ. Elles ont également renforcé mes compétences en résolution de problèmes et m'ont poussé à être plus créatif et autonome dans mes recherches et mes solutions. Malgré ces défis, ces expériences ont été extrêmement formatrices et ont grandement contribué à mon développement en tant que développeur.

Remerciements

Je tiens à exprimer ma profonde gratitude envers l'école EFREI Paris pour m'avoir permis d'étudier dans des conditions exceptionnelles cette année. L'environnement académique stimulant et le soutien constant des enseignants ont été essentiels pour mon développement personnel et professionnel. Grâce à cette école, j'ai pu acquérir des connaissances solides et des compétences précieuses qui m'ont préparé à relever les défis du monde professionnel.

Je souhaite également remercier chaleureusement ADventori pour m'avoir accueilli en alternance. L'expérience que j'ai vécue au sein de cette entreprise a été extrêmement enrichissante. J'ai bénéficié de nombreux conseils et retours constructifs qui ont grandement contribué à l'avancement de mon projet de fin d'année.

Je tiens particulièrement à remercier Thomas Da Costa, mon tuteur, pour son encadrement et ses précieux conseils tout au long de mon parcours chez ADventori. Sa disponibilité et son expertise m'ont été d'une aide inestimable.

Enfin, un grand merci à Pierre Antoine Durgeat, le patron de l'entreprise, pour m'avoir offert cette opportunité et pour son soutien constant. Sa vision et son leadership ont créé un environnement de travail inspirant et motivant.

Conclusion

Mon projet de développement de l'application web Uber Cut a été une expérience enrichissante.

J'ai utilisé le Framework Symfony PHP pour créer une application web robuste et conviviale, répondant aux besoins de mes utilisateurs.

Au cours du projet Uber Cut, j'ai rencontré plusieurs défis techniques, mais j'ai surmonté ces difficultés et créé une application web qui est à la fois fonctionnelle et, je l'espère, élégante. Uber Cut m'a beaucoup apporté d'un point de vue technique et notamment du point de vue de la conception d'application. J'ai également progressé d'un point de vue méthodologique, car j'ai réalisé que le manque de rigueur peut rapidement entraîner des retards considérables.

Je suis fier du travail accompli et convaincu que mon application web de gestion de trajets pourrait être un outil précieux pour mes utilisateurs, facilitant leurs déplacements quotidiens.

Pour la suite, je ne compte pas laisser Uber Cut de côté. Je prévois de continuer à améliorer ce site et à lui apporter autant de fonctionnalités que possible pour répondre aux besoins évolutifs des utilisateurs de ce type d'application et enfin proposer une application entièrement fonctionnelle d'ici janvier 2025.

Je suis impatient de voir comment Uber Cut va évoluer et d'explorer toutes les opportunités d'amélioration et d'innovation pour faire de cette application un outil indispensable pour la gestion de trajets.